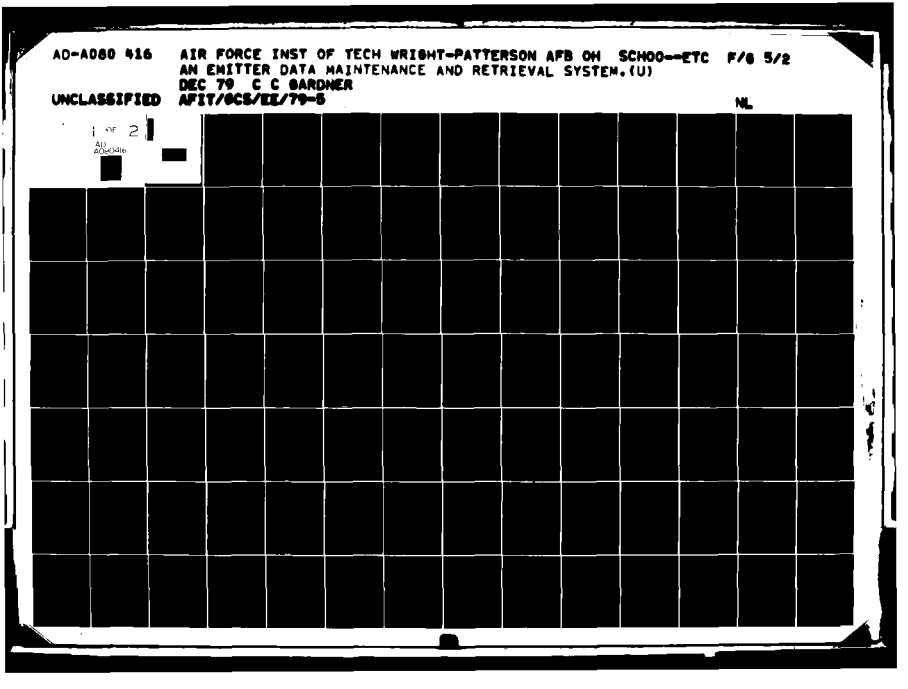
AD-A080 416 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 5/2
AN EMITTER DATA MAINTENANCE AND RETRIEVAL SYSTEM.(U)
DEC 79 C C GARDNER
UNCLASSIFIED AFIT/GCS/EE/79-5 NL



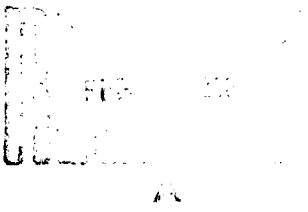
AFIT/GCS/EE/79-5

(6) AN EMITTER DATA MAINTENANCE
AND RETRIEVAL SYSTEM,

(9) Master's - THESIS,

(14) AFIT/GCS/EE/79-5 (10) Clifford C. Gardner
Capt USAF

(11) Dec '79 (12) 152



Approved for public release; distribution unlimited

012 225

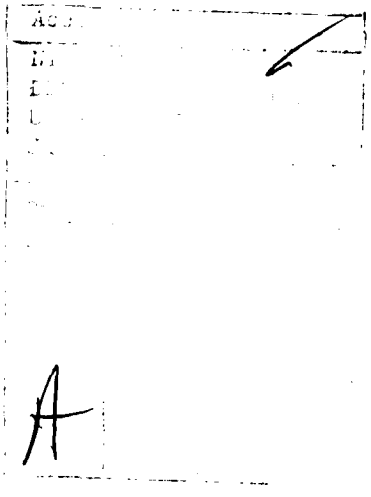
MT

AFIT/GCS/EE/79-5

# AN EMITTER DATA MAINTENANCE AND RETRIEVAL SYSTEM

THESIS

Presented to the Faculity of the School of Engineering
of the Air Force Institute of Technology
Air Training Command
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science



by
Clifford C. Gardner, B.S.
Capt USAF
Graduate Computer Systems

December 1979

Approved for public release; distribution unlimited.

## Preface

The initial proposal to design a system for maintaining data on electronic emitters, was made by Mr. William McQuay of the Electronic Warfare Analysis branch of the Air Force Avionics Laboratory (AFAL/WRA). This project provided an opportunity to evaluate many software engineering techniques being proposed to improve the quality of computer programs.

Mr. McQuay acted as user liaison for this project. He helped define the tasks and scope for the system and patiently provided the technical information necessary to implement it. Thanks also goes to Dr. Thomas Hartrum who acted as faculty advisor for the thesis. Technical advice was received from Major James P. Rutledge who, with Prof Charles Richard, also provided constructive criticism of this document. Very special thanks goes to my wife, Julie, for support and patience during the course of this project.

Clifford C. Gardner

Contents


Volume I PAGE

Preface . . . . . ii

List of Figures . . . . . iv

List of Tables . . . . . vii

Abstract . . . . . ix

I. Introduction . . . . . 1

II. Requirements Analysis . . . . . 6

Introduction . . . . . 6
Design Constraints . . . . . 7
Functional Specifications . . . . . 8
Preliminary Design Decisions . . . . . 14

III. System Design . . . . . 18

Introduction . . . . . 18
Structure Charts . . . . . 18
Summary . . . . . 81

IV. System Data Structures . . . . . 82

V. System Algorithms . . . . . 111

VI. Maintenance Programming Considerations . . . . . 115

VII. Conclusions and Recommendations . . . . . 122

Bibliography . . . . . 124

Appendix A: Structured Analysis and Structures Design Techniques . 125

Appendix B: User's Guide . . . . . 129

Volume II

Program Listing

## List of Figures


| Figure | | Page |
|---|---|---|
| 1 | Electronic Warfare Simulation Analysis | 2 |
| 2 | System Overview | 9 |
| 3 | Major System Functions | 10 |
| 4 | Process Control Cards | 12 |
| 5 | The Update Function | 13 |
| 6 | The Produce Function | 15 |
| 7 | The Search Function | 16 |
| 8 | System Structure | 19 |
| 9 | Initialize | 24 |
| 10 | Get Command | 28 |
| 11 | Get Card | 31 |
| 12 | Update | 34 |
| 13 | Delete | 38 |
| 14 | Insert Update Record | 41 |
| 15 | Insert | 44 |
| 16 | Change | 48 |
| 17 | Merge | 52 |
| 18 | Process Update Record | 56 |
| 19 | Search | 59 |
| 20 | Get Search Keys | 63 |
| 21 | Put Matching Emitters | 66 |
| 22 | Get Output Formal | 69 |
| 23 | Produce | 72 |
| 24 | Get Requested Emitter Numbers | 75 |

List of Figures (Con't)


| Figure | | Page |
|---|---|---|
| 25 | Put Produce Data File | 78 |
| 26 | Emitter Data File Records | 84 |
| 27 | Source Record | 88 |
| 28 | Transmitter Record | 89 |
| 29 | Receiver Record | 90 |
| 30 | PRF Record | 91 |
| 31 | PW Record | 92 |
| 32 | Beam Record | 93 |
| 33 | Nodrate Record | 95 |
| 34 | RF Record | 96 |
| 35 | Scan Record | 97 |
| 36 | Output Type Record | 99 |
| 37 | Update Record | 102 |
| 38 | Modification Record | 103 |
| 39 | Chain Record | 104 |
| 40 | Query Master Record | 105 |
| 41 | Query Record | 107 |
| 42 | Match Record | 108 |
| 43 | Emitter Number Record | 109 |
| 44 | Chain Record | 110 |
| 45 | Structured Analysis Syntax | 126 |
| 46 | Module to Module Information Flow | 126 |
| 47 | Information Arrows | 128 |

List of Figures (Con't)


| Figure | | Page |
|---|---|---|
| 48 | Decision and Iteration Syntax . . . . . | 128 |
| 49 | Update Control Cards . . . . . | 131 |
| 50 | Delete Data Cards - One Value per Card . . . . . | 131 |
| 51 | Delete Data Cards - Multiple Value per Card . . . . . | 131 |
| 52 | Output Selection Control Cards . . . . . | 133 |
| 53 | Search Control Cards . . . . . | 137 |
| 54 | Produce Control Cards . . . . . | 139 |
| 55 | Sample Control Stream . . . . . | 139 |

## List of Tables


| Tables | | Page |
|---|---|---|
| I | Parameters for System Structure | 20 |
| II | Parameters for Initialize | 25 |
| III | Parameters for Get Command | 29 |
| IV | Parameters for Get Card | 32 |
| V | Parameters for Update | 35 |
| VI | Parameters for Delete | 39 |
| VII | Parameters for Insert Update Record | 42 |
| VIII | Parameters for Insert | 45 |
| IX | Parameters for Change | 49 |
| X | Parameters for Merge | 53 |
| XI | Parameters for Process Update Record | 57 |
| XII | Parameters for Search | 60 |
| XIII | Parameters for Get Search Keys | 64 |
| XIV | Parameters for Put Matching Emitters | 67 |
| XV | Parameters for Get Output Format | 70 |
| XVI | Parameters for Produce | 73 |
| XVII | Parameters for Get Requested Emitter Numbers | 76 |
| XVIII | Parameters for Put Produce Data File | 79 |
| XIX | Fields of the Emitter Data Record | 85 |
| XX | Fields for Source Record | 88 |
| XXI | Fields for Transmitter Record | 89 |
| XXII | Fields for Receiver Record | 90 |
| XXIII | Fields for PRF Record | 91 |
| XXIV | Fields for PW Record | 92 |

List of Tables (Con't)


| Tables | | Page |
|---|---|---|
| XXV | Fields for Beam Record | 93 |
| XXVI | Fields for Nodrate Record | 95 |
| XXVII | Fields for RF Record | 96 |
| XXVIII | Fields for Scan Record | 97 |
| XXIX | Fields for Output Type Record | 99 |
| XXX | Number to Character Conversion Table | 100 |
| XXXI | Character to Number Conversion Table | 100 |
| XXXII | Fields for Update Record | 102 |
| XXXIII | Fields for Modification Record | 103 |
| XXXIV | Fields for Chain Record | 104 |
| XXXV | Fields for Query Master Record | 105 |
| XXXVI | Fields for Query Record | 107 |
| XXXVII | Fields for Match Record | 108 |
| XXXVIII | Fields for Emitter Number Record | 109 |
| XXXIX | Fields for Chain Record | 110 |
| XL | Alphanumeric Work Variables Affected by Conversion | 119 |
| XLI | Update Input and Produce Output Formats - Data | 133 |
| XLII | Legal Input/Output Data Types | 134 |
| XLIII | Legal Search Keys | 137 |

## Abstract

The Electronic Warfare Analysis branch of the Air Force Avionics Laboratory uses several computer models to simulate electronic environments. One of the inputs to these models is a file of parameters for electronic emitters. This file is currently compiled and maintained manually. The large volume of data available makes this an inefficient method.

This document describes the design and implementation of a system which processes the electronic emitter data. It provides a means of updating the data file with the latest information, producing a file as input to the simulation models, and searching the data file for emitters possessing certain key characteristics. Several software engineering techniques were used for this project. Structured Analysis was used to define the requirements. Structured Design was used to modularize the system. A block structured language was used to code the system and a structured testing method was utilized to provide an easily maintainable product.

# I. Introduction

To generate a realistic simulation of a large air defense system, a significant data base of information concerning radar emitters must be maintained and accessed. This paper describes the design and implementation of an information maintenance and retrieval system which provides this capability. The objective of this thesis was to provide a means for the efficient manipulation of the large amounts of electronic emitter data. Automation of this data handling will significantly reduce the workload of the user.

Top-down development and testing techniques were used for this project. Structured analysis conventions were used to define the requirements and structured design techniques were used to organize the tasks into easily coded and maintainable modules.

The remaining sections of this introductory chapter provide background material on the utility of this information handling system, define the problem specifically attacked for this project, describe the approach taken toward the solution of the problem, state the scope of this study, and present an overview of the remaining chapters of this thesis.

## Background

The Air Force Avionics Laboratory (AFAL) utilizes several large air defense models for its electronic warfare analysis. These simulation

programs run on dual Itel computers at the ASD Computer Center. The models are run several times a year using various scenarios. The preparation for each run involves a large amount of manual labor. Figure 1 represents the total system.

One of the inputs that is prepared for these models is a radar parameters file. This is a file containing data on the electronic emitters which will be used in the scenario designed for the particular simulation run. For each such emitter, the file contains parameters such as scan rate, pulse width, and transmitter power. The model uses these inputs to simulate the electronic environment of a particular region, real or fictitious. A large variety of emitters and associated parameters is necessary to provide a realistic result. The variety is currently limited by the manner in which the data is stored and retrieved.

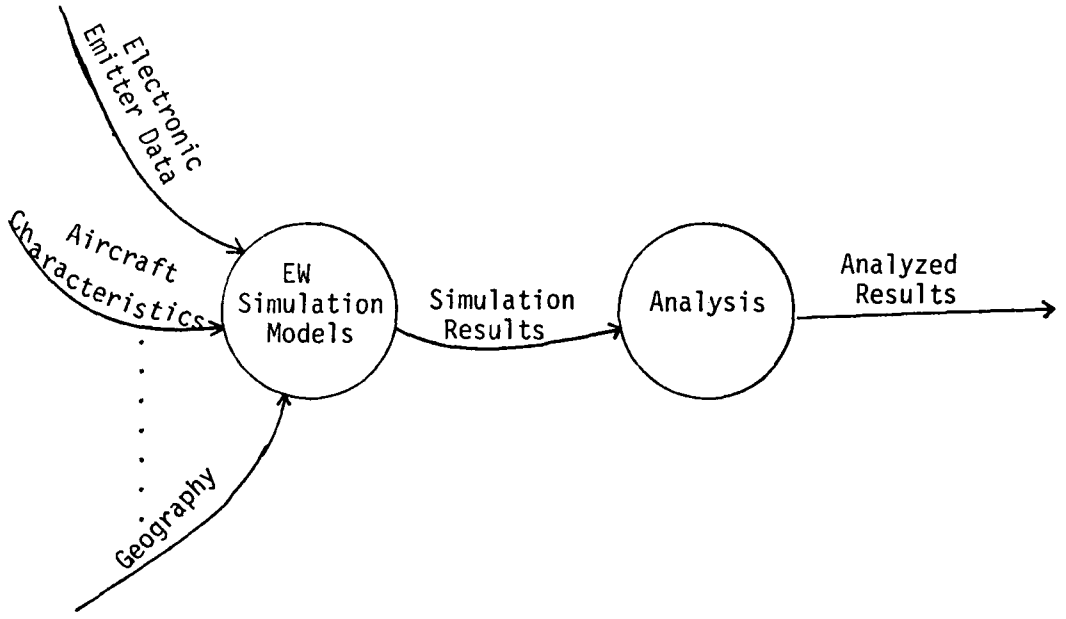



FIGURE 1. Electronic Warfare Simulation Analysis

## Problem Description

Prior to the implementation of this system, the parameters for all maintained emitters were stored as a sequential file of punched computer cards. For each run of the model, this file was manipulated by hand to produce a second file which was used as input to the simulation programs. The user selected the emitters he wished to use and filled out a worksheet, choosing the parameters from those associated with that equipment in the card file. If the user desired to simulate equipment with certain characteristics, but did not know which emitters possessed those features, he searched the entire file by hand to determine whether such equipment existed. Since the information was stored in such a unwieldy manner, it was not always updated with the latest technical and intelligence data. The user checked the data on his worksheet against the most current information as he was preparing for the run, slowing the preparation process still further. Once the worksheets were prepared and verified, the data was repunched onto cards and loaded onto a tape which was compatible with the air defense models. The preparation for one computer run required six man-months for the radar parameters alone.

Due to the volume of the card file containing the radar characteristics (30-60 cards per emitter), data on only 400 emitters was maintained. The analysis branch of the AFAL has intelligence data on over 17,000 emitters. Two thousand of these are needed as inputs to the models. Additional information such as the source of the technical data, date of last update, classification, and comments was not maintained but should have been stored with the radar parameters.

## Approach

The solution to the problem described above is an automated system to maintain a file of electronic emitter data, search that file for specific information, and produce a file of emitter data to be used in the simulation run of the air defense models. The development of this system was accomplished by working with the user to define the exact requirements; breaking those requirements down into tasks and sub-tasks which could be modularized; coding these modules in a clear, maintainable manner; testing the system in a top-down manner; demonstrating the system for the user; and documenting the system thoroughly.

## Scope

By attacking the specific problem involving the radar parameters, a significant contribution to the AFAL mission is achieved. This project solves the problem in a manner which can be adapted or expanded to address similar problems in the future. This paper provides documentation that will assist maintenance programmers in such an expansion or adaptation effort, as well as providing complete user documentation.

This project resulted in a system which accomplishes three tasks. First, the system maintains a file of all desired emitters with their parameters. It will allow data to be easily updated or deleted as new technical information becomes available. It allows a search by the user on several key values and produces a listing of all emitters possessing the desired characteristics. Finally, it produces a tape, which is compatible with the air defense models, which contains all user-selected parameters.

## Overview of the Thesis

This paper provides documentation on the requirements analysis, system design, and software implementation for this system. It also contains a user's guide and maintenance programming documentation.

Chapter II presents the requirements analysis documentation associated with this project. Chapter III contains the system design information, including the structure charts defining the tasks and subtasks in modular form. Chapter IV describes the data and its organization. Chapter V defines the major algorithms used to produce the desired results. Chapter VI contains the additional detailed information which should facilitate program expansion, adaptation, or conversion. Chapter VII summarizes the project and suggests possible enhancements. The appendices to this thesis describe the software engineering techniques employed, provided a listing of the actual code, and contain the user's guide with sample inputs and outputs for the system.

# II. Requirements Analysis

## Introduction

The Requirements Analysis phase of the software development process is often neglected for small projects, but it can be a very important step. It impels the designer to set down, in some form, a description of what the proposed system is to do. This produces a dialogue between the user and the designer that often turns up misunderstandings and communication breakdowns before too much effort has been expended producing a design that cannot produce the desired results.

The requirements analysis for this project concerned four areas: context analysis, design constraints, Structured Analysis, and preliminary design decisions. The context analysis concerns the position of the desired system in the overall configuration in which it must operate. This was discussed in the background section of Chapter 1. The design constraints state the assumptions that were made at the outset of the project and the guidelines that were set by the user concerning the environment under which the system must operate. These are discussed in the next section of this chapter. The following section contains a Structured Analysis breakdown of the project. It discusses the technique used and describes the requirements at a high level. The last section contains a discussion of the design tradeoffs that could be made at this point based on the contraints and requirements.

Design Constraints

Several constraints were placed, at the outset of the project, on the system to be developed. These concern the data to be processed and the environment in which the program must operate. These conditions together with a general statement of the tasks to be performed, defined the inputs to the requirements analysis phase.

The first goal was to produce a means of maintaining the information that was available on several thousand electronic emitters. Details on the type of information to be maintained are provided in Chapter IV. Several constraints arose during the analysis of this task. First, much of the data would be classified. The user also suggested that the system should run on the CDC/Cyber computer system at the ASD Computer Center, until a new secure, dedicated system became available at AFAL. Classified jobs on the CDC system are run on the third shift. This mandated a batch system and led to a decision to store the master data base on a magnetic tape because classified information cannot be stored on disk, and other forms of storage would prove too bulky. The impending conversion to a new, dedicated system also provided a constraint which impacted on future design decisions.

Another task of the program was to produce a problem tape which contained the characteristics of requested electronic emitters. The output was specified as a magnetic tape because it was to be used as an input to the air defense models. These models also expected a fixed format for the data.

The third task was to produce a hardcopy listing of all electronic

emitters with certain desired characteristics. This imposed no constraints beyond the goal itself.

One last goal was imposed to the designer by himself. This was to produce a system which met the fundamental goals of software engineering, efficiency, reliability and maintainability.

## Functional Specifications

The diagrams used to express the system requirements are patterned after the Data Flow Diagrams of Demarco' (Ref 4:47-89). A brief discussion of this structured analysis technique is contained in Appendix A. Only diagrams depicting the higher levels of the system were produced. This was quite adequate for a project of this size.

Figure 2 represents the overall system to be developed. The inputs are a series of control cards and the master data file. The outputs are a problem tape, a hardcopy listing, and/or a new, updated data file. The process shown is then broken down into the major tasks of the system in Diagram 0.

Figure 3 depicts this base or 0 level diagram. Process 1 handles the control cards, reports errors in the control card stream, and passes control to the task that is requested next. Process 2 represents the maintenance function. It accepts as input the update control cards and the current emitter data file. It produces a new data file incorporating all legal, requested changes. It also reports any errors discovered in the update attempt. Process 3 produces the problem tape to be input to the air

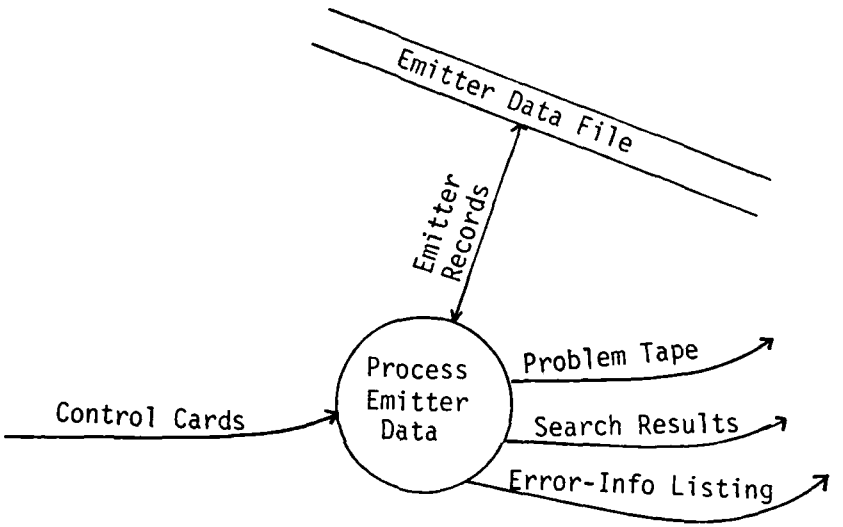



Figure 2. Diagram -1: System Overview

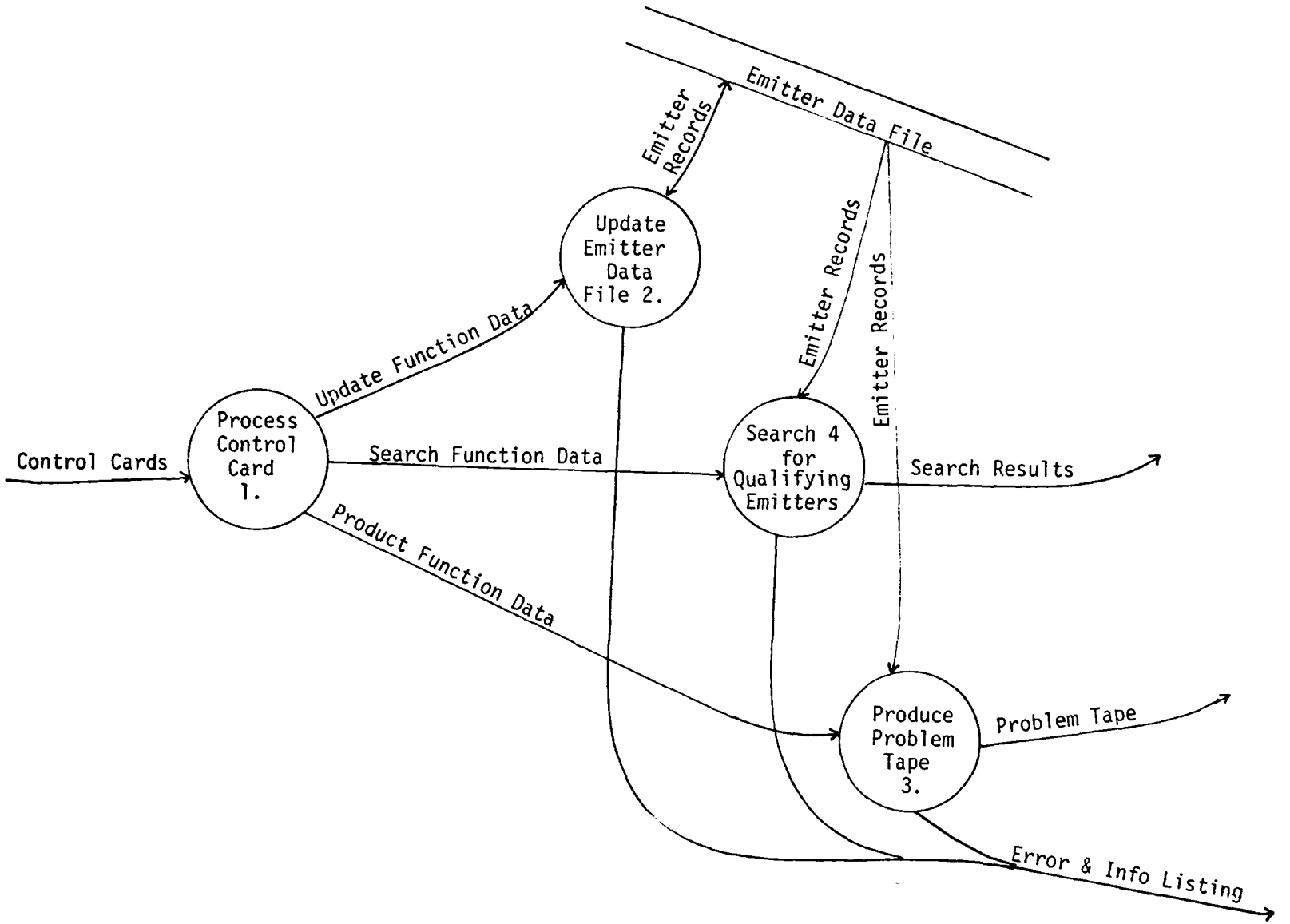



Figure 3. Diagram 0: Major System Functions

defense models. It also uses control cards and the emitter data file as inputs and reports the errors it discovers. Process 4 searches the data file on requested keys. It produces a listing of all emitters possessing the requested characteristics, as well as all errors found, from inputs of control cards and the latest emitter data file. Processes the subsections below. At the conclusion of each task the control reverts to Process 1 which calls the next task or terminates the processing.

Control Card Handler. The processing of control cards is described in Figure 4. It consists of 3 steps. A control card is read. It is then edited to determine if it contains a legal command. If it does not contain a legal task, it is rejected, an error message is output, and new control cards are read and edited until a legal command is found or until the control card stream is terminated. Once a legal command has been found the system passes control to the process that has been requested.

Update Function. The update function performs the data file maintenance task. Figure 5 illustrates the subtasks of this function. Process 2.1 handles the control cards for this subtask in a similar manner to the main control card handler described above. The control cards are read, edited, and interpreted then control is passed to the next process. The next process (2.2 - 2.4) form update records that either change an existing record, delete a current record, or insert a new record. After all update records have been read and interpreted, they are passed to the sort and merge processes.

The sort process will arrange the records in order based on the same key that organizes the master emitter data file. The update records are then merged into the data file and the new emitter data file is written

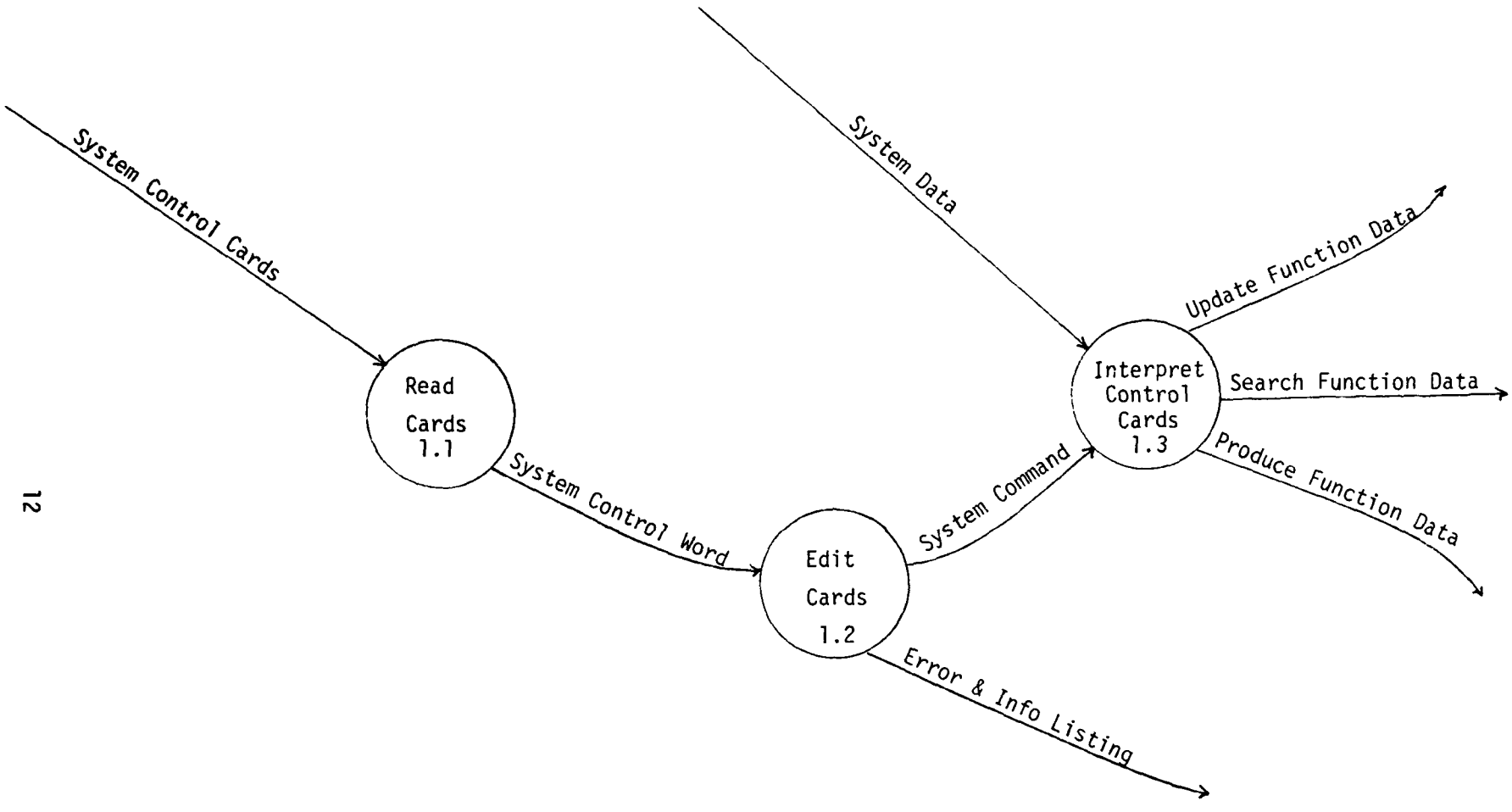



Figure 4. Diagram 1; Process Control Cards

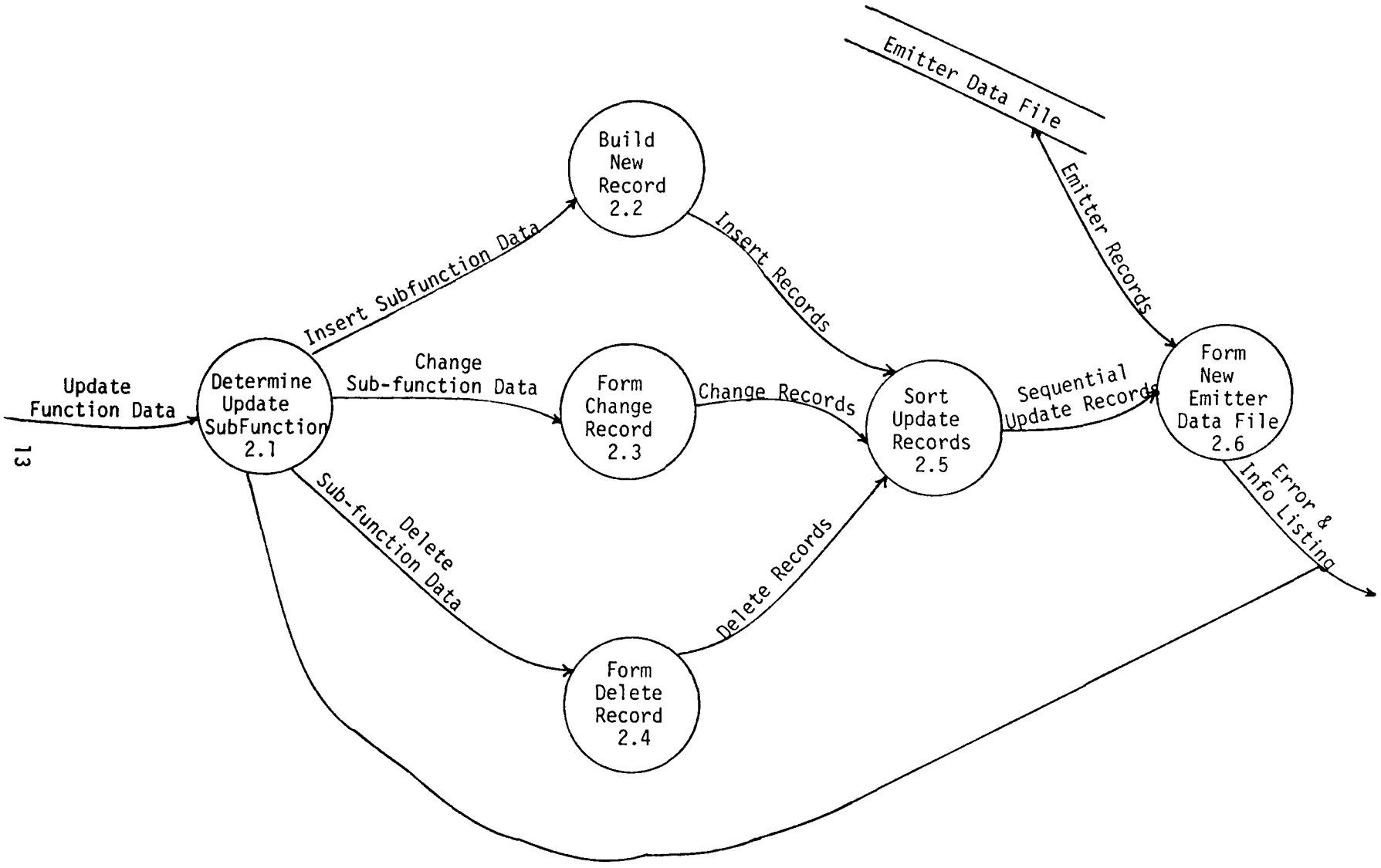



Figure 5. Diagram 2: The Update Function

back for future use. The other output of this function is a listing of input errors and processing information.

Produce Function. The problem tape containing electronic emitter data for the air defense models is the major output of the produce function. As depicted in Figure 6, the control cards requesting the emitters for the tape are read, edited, and sorted by the first process (3.1). The requested records are then retrieved from the emitter data file. The specific characteristics to be output for the requested emitters are from the retrieved records and written onto the problem tape in a format that is compatible with the air defense models.

Search Function. Figure 7 is a representation of the last major function of this system. The search function first reads and edits the keys upon which the search is to be made. These keys consist of a characteristic and a value or range that an emitter must match to qualify. These search keys are then used to retrieve the qualifying emitter records from the emitter data file. The characteristics to be output, for these matching records, are then read and used to extract the desired parameters. This information is then listed for the user.

Preliminary Design Decision. A major design decision could be made at this point. This was the choice of programming language. The air defense models with which this program is associated are written in FORTRAN. However, the data manipulation required indicated the need for a language that is more flexible in its use of data structures. Other requirements were for a language that was readable, a prime ingredient in maintainability, and for a language available on the CDC system as well as the follow on system at AFAL, a VAX 11/780. A language that met

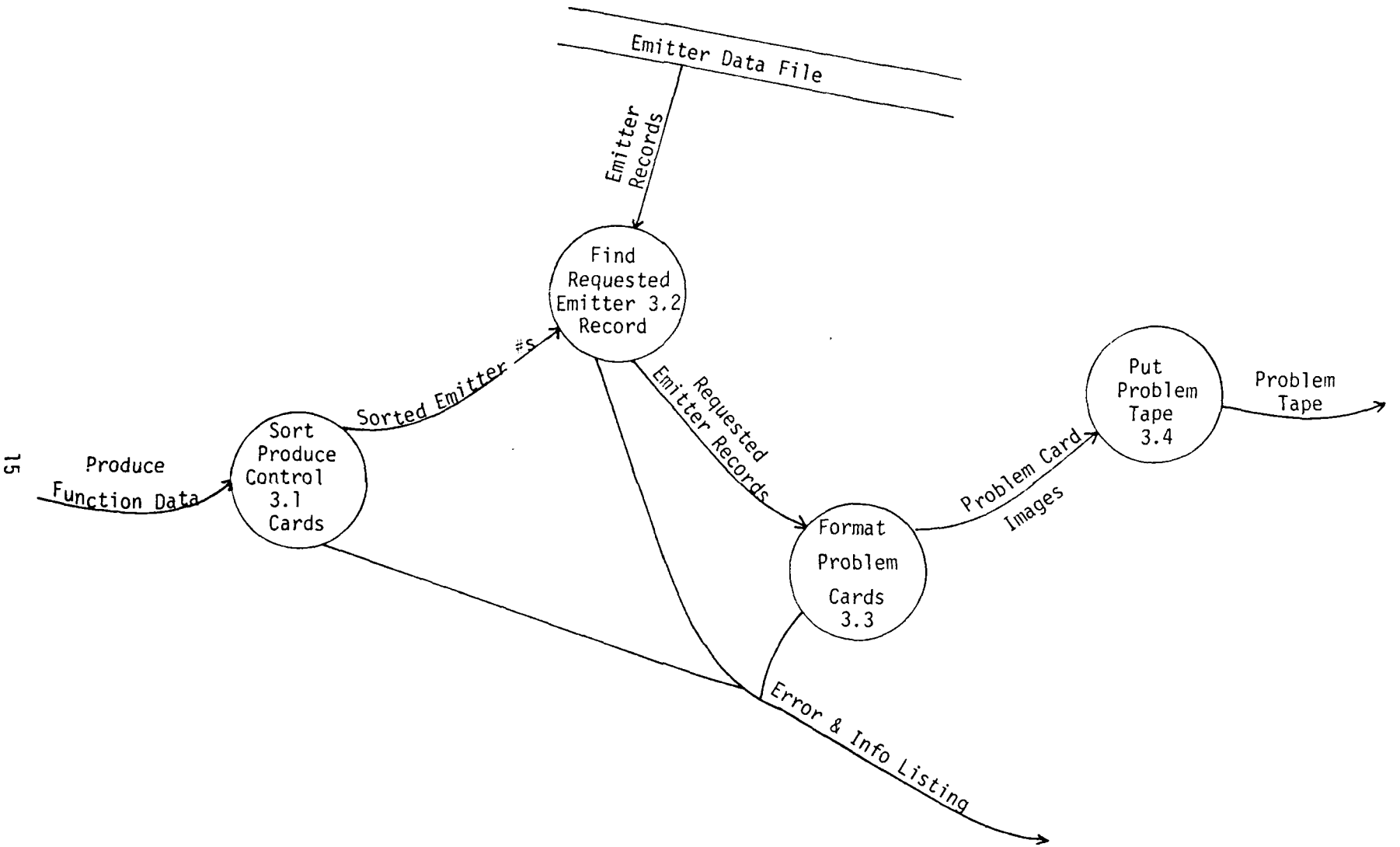



Figure 6. Diagram 3: The Produce Function

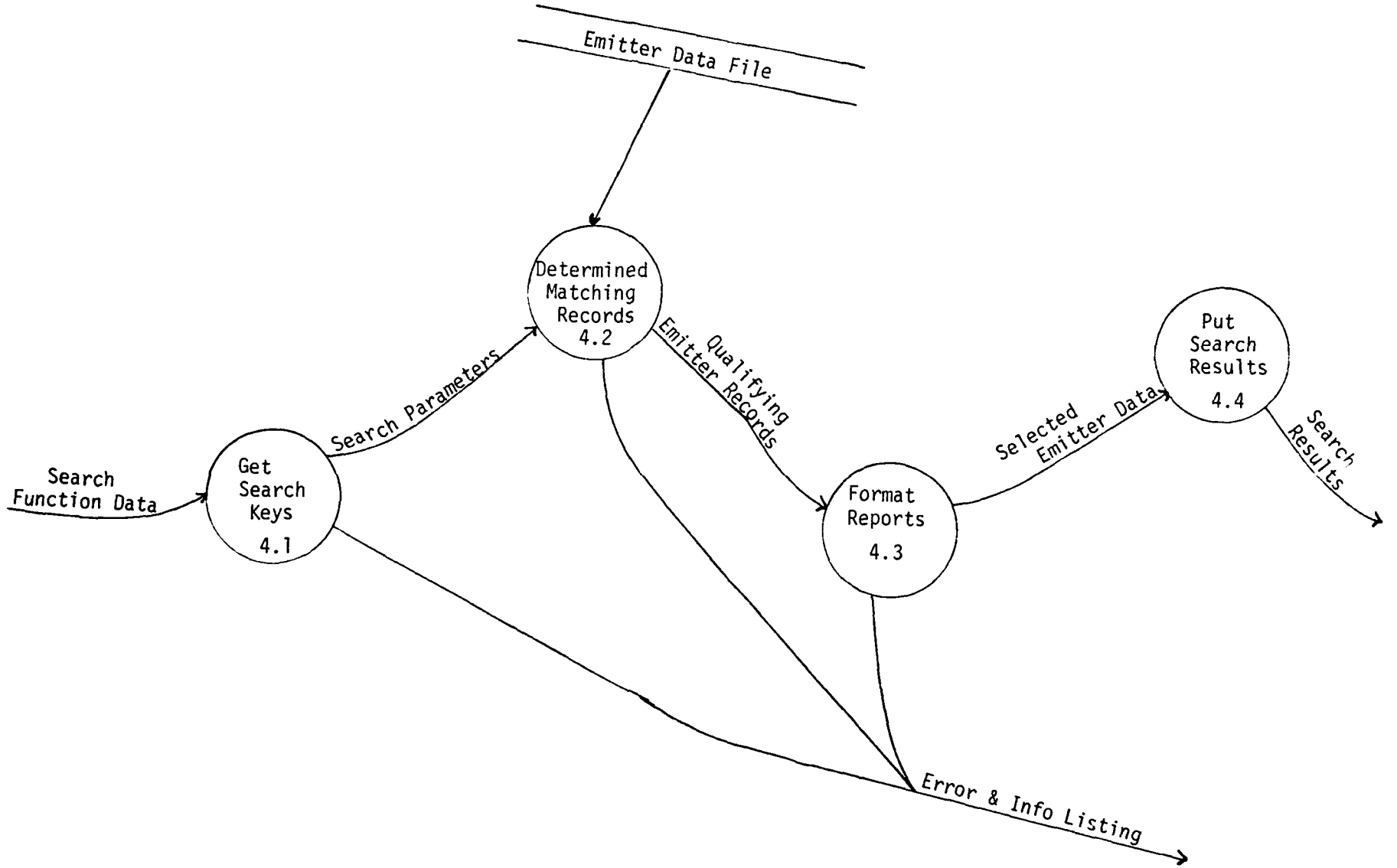



Figure 7. Diagram 4: The Search Function

all these requirements was PASCAL. This choice had a big impact on the design specification discussed in the next chapter. A discussion of the other design decisions is contained in later chapters.

# III. System Design

## Introduction

The purpose of the design phase is to define the functions and operations necessary to satisfy the system requirements. For this thesis a technique known as Structured Design (Ref 8:373-383) was utilized. A brief explanation of this terminology is available in Appendix A.

The first step of the technique is to produce a first cut structure chart from the data flow diagrams produced during the requirements analysis phase. This contains the major functions which have been defined. Normally, this consists of an afferent or input function(s), one or more transform modules and an output or efferent function. As can be seen in the previous chapters, the functions of this system did not break down this clearly, but the general idea remained the same.

These functions are then progressively broken down into lower level structure charts until each module performs a specific task. A review of the charts may then reveal modules which perform no real function. Lower level modules can then be collapsed into these modules, so that all modules perform a necessary, specific task.

## Structure Charts

The structure charts for this system are shown as Figures 8 - 25. They are organized as a series of structure charts, each followed by a table containing the data and control flows and a text description of the modules. Certain modules and branches are described separately to show

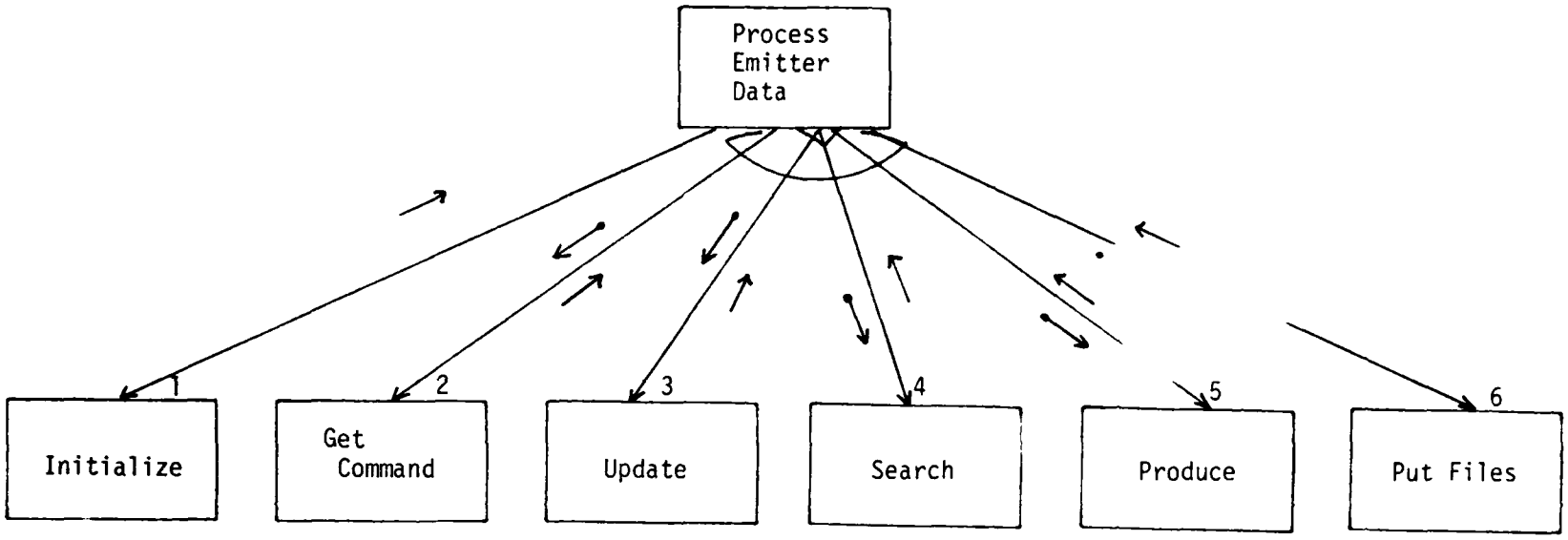



Figure 8. System Structure

TABLE I

Parameters for System Structure

| Input | Output |
|---|---|
| 1. - - - - - | Conversion Tables, Editing Tables, Control Variables |
| 2. Control Cards | System Command, Error Information, End-of-Input |
| 3. Update Control Cards, Emitter Data File | Updated Emitter Data File, Error Information, End-of-Function |
| 4. Search Control Cards, Emitter Data File | Search Results, Error Information, End-of-Function |
| 5. Produce Control Cards, Emitter Data File | Emitter Data Card Images, Error Information, End-of-Function |
| 6. Updated Emitter Data File, Search Results, Emitter Data Card Images, Error Information | Emitter Data File Tape, Listing of Search Results, Problem Tape, Listing of Error Messages |

that they are accessed from various modules and branches. These specific cases are indicated by a letter inside a circle (e.g., (A)).

## Module Descriptions for System Structure

Process Emitter Data. This is the coordinate or executive module. Its function is to control the overall operation of the system. When the executive is entered the first time the initialize routines are invoked to set up tables of use to various functions of the system. Control is then passed to a routine which reads and interprets the system control cards. Control is then passed to the appropriate function. Any output from a function is performed next, followed by an attempt to obtain another system command. This is repeated until no system control cards remain.

Initialize. This is a module which sets certain control variables and tables. These tables contain the legal input and output types as well as tables which facilitate the conversions of numbers to characters and characters to number.



Get Command. The reading, editing, and interpreting of system control cards is performed in this module. Information concerning errors is passed to the executive to be output to the users. The system commands determined here decide which of the major functions is performed next.

Update. Data file maintenance is controlled by this process. The module reads the function control cards and data cards, then updates the master emitter data file. The new file is then made available to the next function and also rewritten to permanent storage on tape.

Search. This module controls the function which examines the emitter data file and extracts records

which have characteristics selected by the user. The user also may specify which information from these records he wishes to have output.

Produce. The produce function extracts records from the emitter data file based on emitter numbers chosen by the user. This information is then used to build a tape of card images to be input to the air defense models. The specific parameters on the tape are also selected by the user.

Put Files. This process represents the general output structure of the system. It provides the user with information and error messages concerning his job as well as the output he requested.

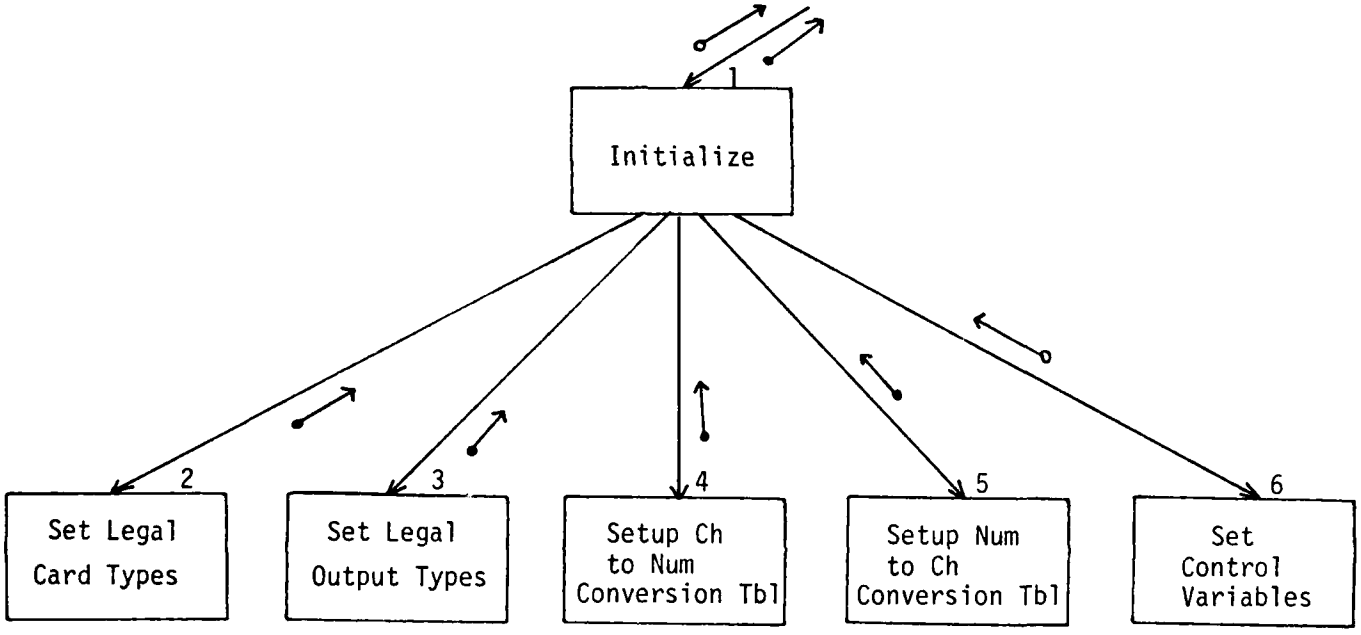



Figure 9. Initialize

TABLE II

Parameters for Initialize

| Input | Output |
|---|---|
| 1. - - - - - | Conversion Tables, Editing Tables, Control Variables |
| 2. - - - - - | Input Card Editing Table |
| 3. - - - - - | Output Type Editing Table |
| 4. - - - - - | Character to Number Conversion Table |
| 5. - - - - - | Number to Character Conversion Table |
| 6. - - - - - | Control Variables |

## Module Descriptions for Initialize Branch

Initialize. The initialization process is controlled in this module. It oversees the setting of various control variables and parameters as well as initializing output to the top of the page.

Set Legal Card Types. This process sets a table which is used by the update function to determine whether an input card is of a legal type. These card types are numbers which indicate which parameter is on a particular data card. A list of the legal card types is contained in the User's Manual (Appendix C). The Produce Function uses the table to produce a card image compatible with the air defense models.



Set Legal Output Type. This process sets a table of legal output types. A list of these alphanumeric types is available in the User's Manual (Appencix C). This table is used by the Search and Produce Functions to determine which output types are requested by the user. The Produce Function also uses it to build a card image compatible with the air defense models.

Set Ch to Num Conversion Table. The input conversions for this system are relatively simple and this table is used to convert a character to a numeric digit rather than take the time to use a library function (e.g., ORD). Entries "0".."9" of the table contain the values 0..9.

Set Num to Ch Conversion Table. This table is used to perform the reverse of the function described above. Entries 0..9 contain the system codes for "0".."9".

Set Control Variables. Several Boolean variables are used to control the executive process. These are set to their proper values here.

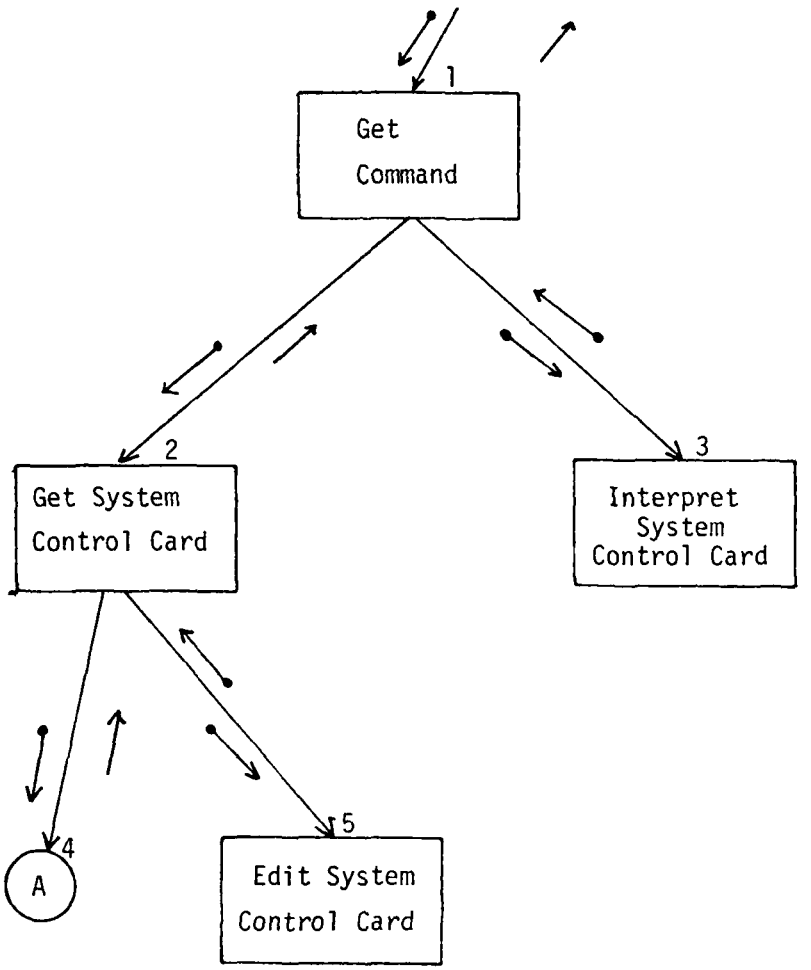



Figure 10. Get Command

TABLE III

Parameters for Get Command

| Input | Output |
|---|---|
| 1. Control Cards | System Command, Error Information, End-of-Input |
| 2. Control Cards | Edited Control Word, Error Information |
| 3. Edited Control Word | System Command |
| 4. Control Cards | Control Word |
| 5. Control Word | Edited Control Word, Error Information |

## Module Description for Get Command Branch

Get Command. This module provides the coordinate module with the next system command from the control card stream. It controls the reading, editing, and interpreting of the control cards. It also returns error indicators to the executive module.

Get System Control Card. This module obtains a legal system control card or sends an error indicator back up the chain. It controls the reading and editing of the card.

(A) Get Card, a module described in the next section.

Edit System Control Card. When a control word has been obtained, this module tests it against legal system control words. If the word is legal, it is passed for intepretation. If not, an error indicator is returned.

Interpret System Control Card. The legal control word is interpreted as a system command by this module. The command is returned to determine which function will be performed next.

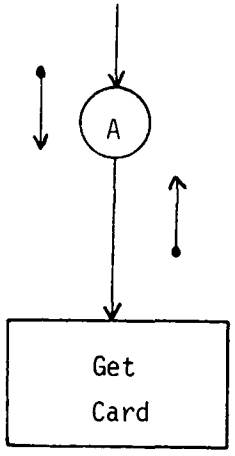



Figure 11. Get Card

TABLE IV

Parameters for Get Card

| Input | Output |
|---|---|
| Control Cards | Control Word |

Used by: Get Command, Update, Get Search Keys, Put Matching Emitters.

## Module Description for the Get Card Branch

Get Card. This module is accessed from several other modules within several different branches, as shown in Table IV. It reads an alphanumeric word in a free format (omitting preceding blanks). It will not however read a word that continues over a line.

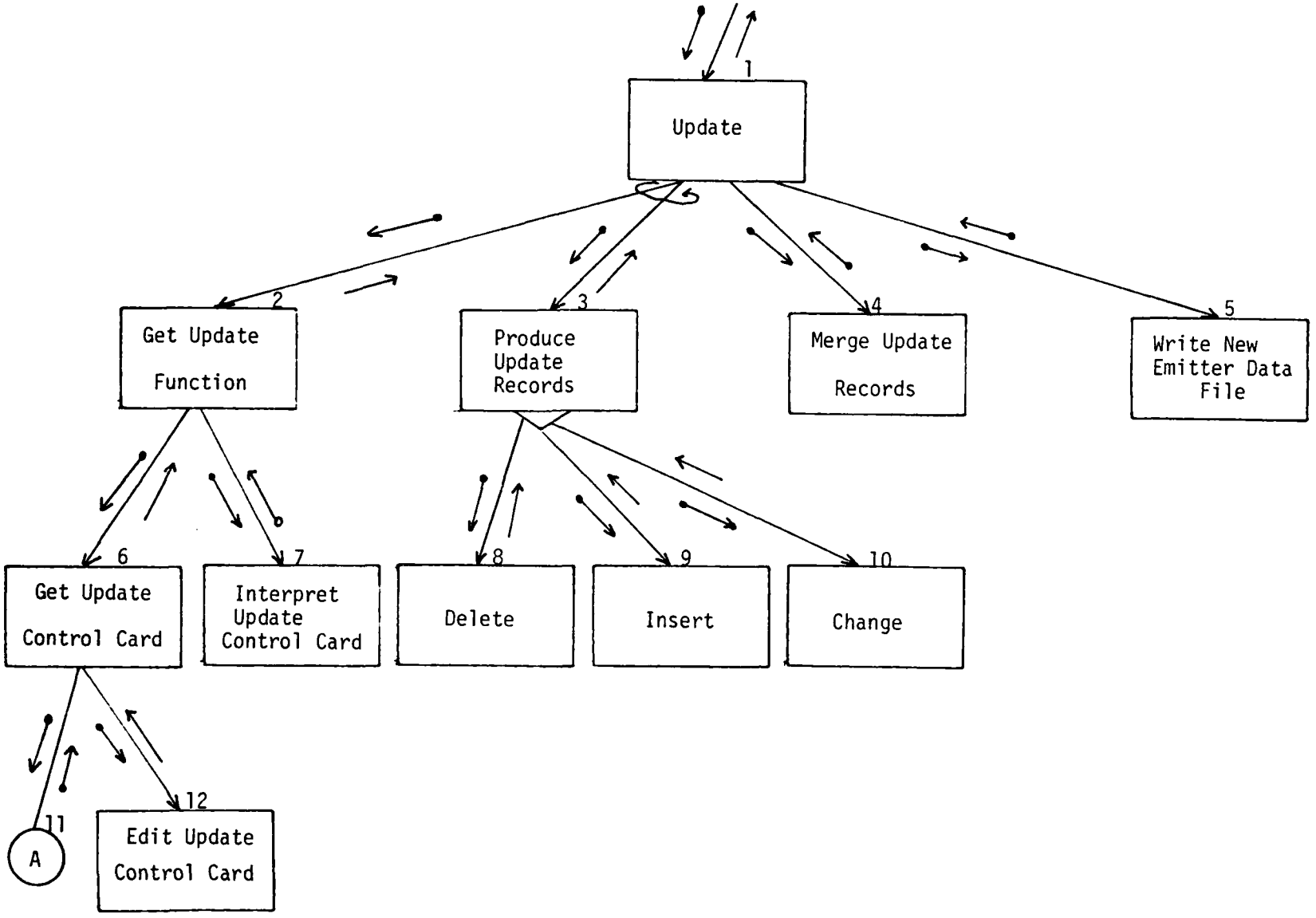



Figure 12. Update

TABLE V

Parameters for Update

| | Input | Output |
|---|---|---|
| 1. | Function Control Cards, Emitter Data File | Updated Emitter Data File, Error Information, End-of-Function |
| 2. | Function Control Cards | Update Subfunction Command, Error Information, End-of-Function |
| 3. | Update Subfunction Command Update Data Cards | Update Records, End-of-Subfunction |
| 4. | Update Records, Emitter Data File | Updated Emitter Data File, Error Information |
| 5. | Updated Emitter Data File | New Master Emitter Data File |
| 6. | Function Control Cards | Edited Control Word, Error Information, End-of-Function |
| 7. | Edited Control Word | Update Subfunction Command |
| 8. | Delete Data Cards | Update Records, End-of-Subfunction |
| 9. | Insert Data Cards | Update Records, End-of-Subfunction |
| 10. | Change Data Cards | Update Records, End-of-Subfunction |
| 11. | Function Control Card | Function Control Word, End-of-Subfunction |
| 12. | Function Control Word | Edited Control Word, End-of-Subfunction |

## Module Description for the Update Branch

Update. This module is the control module for the emitter data file maintenance file. It monitors the input of function control and data cards, the creation of the updated data file, and the output of the new data file for use by subsequent functions. It also returns error and information messages to the user.

Get Update Function. The reading, editing, and interpreting of the update function control cards is controlled by this module. Control is passed here after each update function has been completed. The module returns the next function to be performed, an indicator if no more functions are to be performed, and error information.



Get Update Control Card. This module controls the reading and editing of the update function control cards. It returns an edited control word or error information.

(A) Get Card is described previously as a branch.

Edit Update Control Card. The control word that was input is edited by this module. A legal value is returned if the control word matches those in this module (delete, change, insert) or an error indicator is returned.

Interpret Update Control Card. A legal control word is used by this module to determine which of the

update functions is to be performed next. An indicator is returned to the controlling module.

Produce Update Records. This module performs some initialization and control work before passing control to the selected update function.

Delete. Records are built in this branch that are used to remove records from the emitter data file. The emitter numbers are read and tagged with a deletion indicator. They are then stored in the tree structure discussed in Chapter IV.

Insert. Input data for new records is processed by this branch. The data to be used to build the records is stored in the update record tree.



Change. Individual parameters for existing emitter data records can be modified by this funciton. This branch processes those parameters and stores them in the update record tree structure.

Merge Update Records. The update records which were stored in the previous branches are used by this branch to update the emitter data file. They are merged into the file. Records tagged for deletion are skipped, for change are modified, and for insertion are placed in the proper position.

Write New Emitter Data File. The updated emitter data file is copied to tape for permanent storage and for use by subsequent system functions.

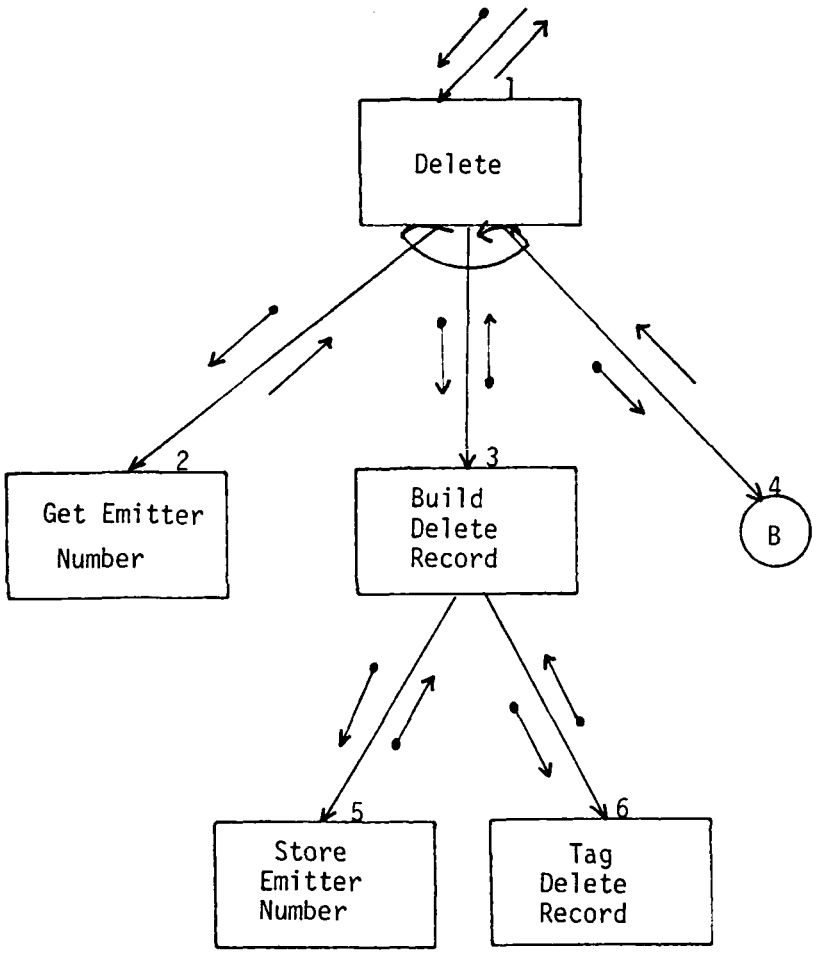



Figure 13. Delete

TABLE VI

Parameters for Delete

| Input | Output |
|---|---|
| 1. Delete Data Cards | Update Records, End-of-Subfunction |
| 2. Delete Data Cards | Emitter Number |
| 3. Emitter Number | Update Record |
| 4. Update Record, Update Record Tree | Update Record Tree, "Unsuccessful Insertion" |
| 5. Emitter Number | Update Record with Emitter Number |
| 6. Update Record with Emitter Number | "Delete" Update Records |

## Module Description for the Delete Branch

Delete. This is the control module for this branch. It reads a series of data cards, processes them, and stores them in the update record tree.

Get Emitter Number. This module reads a value from each card representing a record to be deleted. If the value is numeric, it is returned to the control module, if not, an error message is returned and the card is skipped.

Build Delete Record. The emitter number that was returned from the previous module is stored in an update record (described in Chapter IV). This process is controlled by this module, which creates the new update record.



Store Emitter Number. This module stores the emitter number in the proper position within the update record.

Tag Delete Record. This module tags the update record as a delete record.

(B) Insert Update Record, a module described in the next branch.

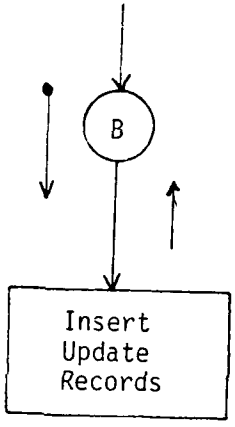



Figure 14. Insert Update Record

TABLE VII

Parameters for Insert Update Record

| Input | Output |
|---|---|
| Update Record, Update Record Tree | Update Record Tree, "Unsuccessful Insert" |

Called by: Delete, Insert, Change

## Module Description for the Insert Update Record Branch

Insert Update Records. This module recursively searches through a binary tree structure and stores an update record at the proper node. An inorder traversal of the structure results in an ordered list of update records. This module is used by each of the update functions.

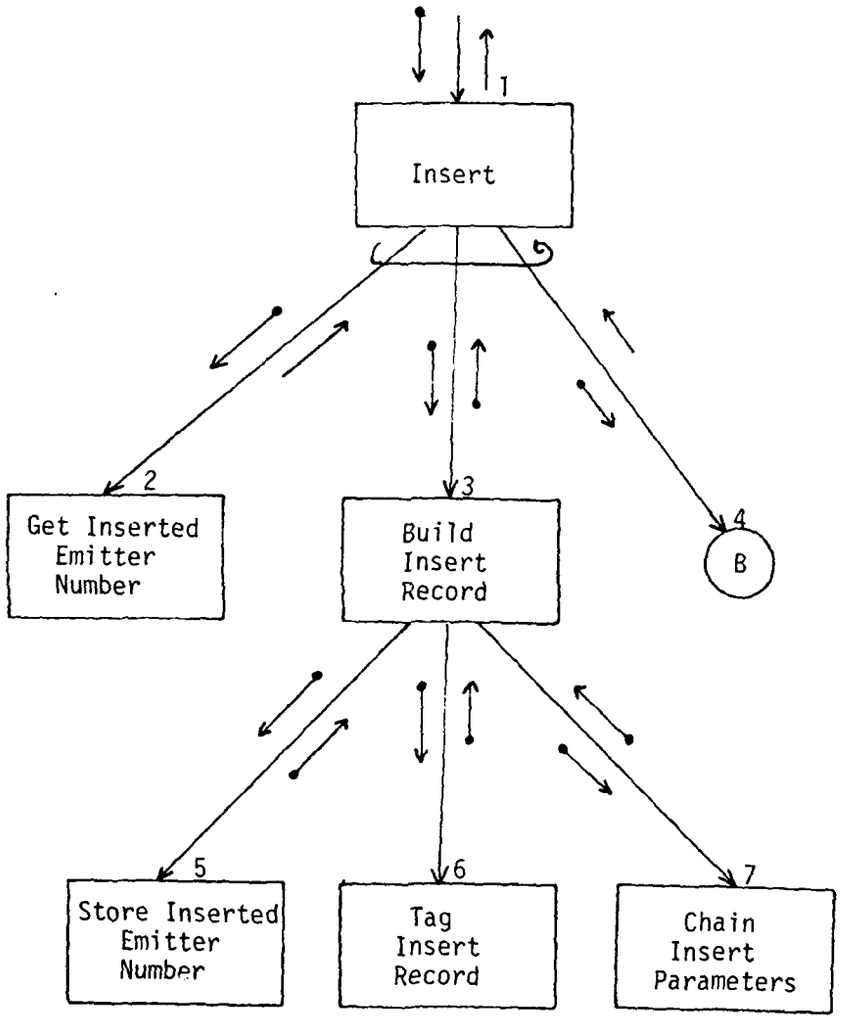



Figure 15. Insert

TABLE VIII

Parameters for Insert

| Input | Output |
|---|---|
| 1. Insert Data Cards, Update Record Tree | Update Records, End-of-Subfunction |
| 2. Insert Data Cards | Emitter Numbers, Insert Parameters, End-of-Subfunction |
| 3. Emitter Numbers, Insert Parameters | Update Records |
| 4. Update Records, Update Record Tree | Update Records, "Unsuccessful Insert" |
| 5. Emitter Number | Update Record with Emitter Number |
| 6. Update Record with Emitter Number | Tagged Update Record |
| 7. Tagged Update Record, Insert Parameters | Update Record |

## Module Description for the Insert Branch

Insert. This module controls the process of reading the data cards and forming the update record which will be used to create a new emitter data record. It then stores the record until it can be merged into the emitter data file.

Get Insert Emitter Number. The emitter number for the new emitter data record is read by this module. If a new numeric value is present, it is passed to the control module. If the value is the same as the last value obtained, an indicator is passed to show that this data card contains additional parameters for the previous emitter. Error information is also sent to the control module.



Build Insert Record. This module controls the process of forming the update record which will be used to insert a new record into the emitter data file. It creates the new update record.

Store Emitter Number. The emitter number obtained from the data cards is stored in the update record by this module.

Tag Insert Record. The update record is tagged as an insertion.

Chain Insert Parameters. Additional parameters are obtained from subsequent data cards and chained to the update record in a linked format. All parameters for the new record are chained in this

fashion. Details of this method are available in Chapter IV.

(B) Insert Update Record, a module in a previously described branch.

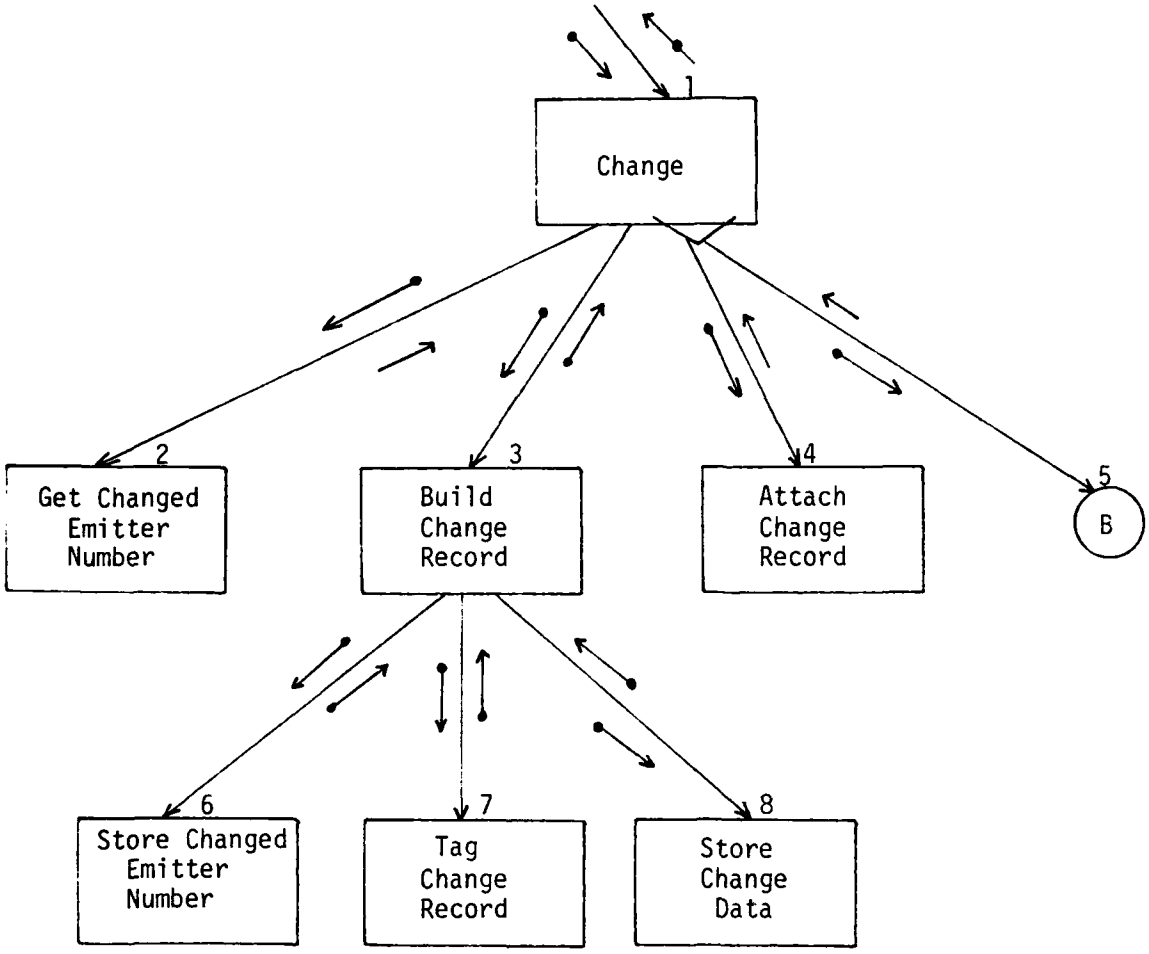



Figure 16. Change

TABLE IX

Parameters for Change

| Input | Output |
|---|---|
| 1. Change Data Cards, Update Record Tree | Update Record Tree, End-of-Subfunction |
| 2. Change Data Cards | Emitter Number, Change Parameter, End-of-Subfunction |
| 3. Emitter Number, Change Parameter | Update Records |
| 4. Update Records, Update Record Tree | Update Record Tree, "Unsuccessful Attach" |
| 5. Update Records, Update Record Tree | Update Record Tree, "Unsuccessful Insert" |
| 6. Emitter Number | Update Record with Emitter Number |
| 7. Update Record with Emitter Number | Tagged Update Record |
| 8. Tagged Update Record | Update Record |

## Module Description for the Change Branch

Change. This module controls the input, construction, and insertion of records that will modify existing emitter data records. It returns error and information messages to the user.

Get Change Emitter Number. This module obtains the emitter number of the record to be modified and checks to see if it is a legal numeric value.

Build Change Record. The update record, which will be used to modify an emitter data record, is built under the control of this record. It creates the new update record.



Store Emitter Number. The emitter obtained from the data card is stored in the update record by this module.

Tag Change Record. The update record being created is tagged as a change record by this module.

Store Change Data. The parameters on the data card are stored in the update record by this module.

Attach Change Record. This module attempts to chain the parameters for this change to an update record already in the update record tree. It searches the tree for an update record with an emitter number which is equal to the emitter number of this update record. If such a record is found, the parameter from this change is chained to the parameters already present in the tree. If no match

is found, an indicator is returned to the change control module which call the insertion module (B), which adds the update record to the tree.

(B) Insert Update Record; a module in a previously described branch.

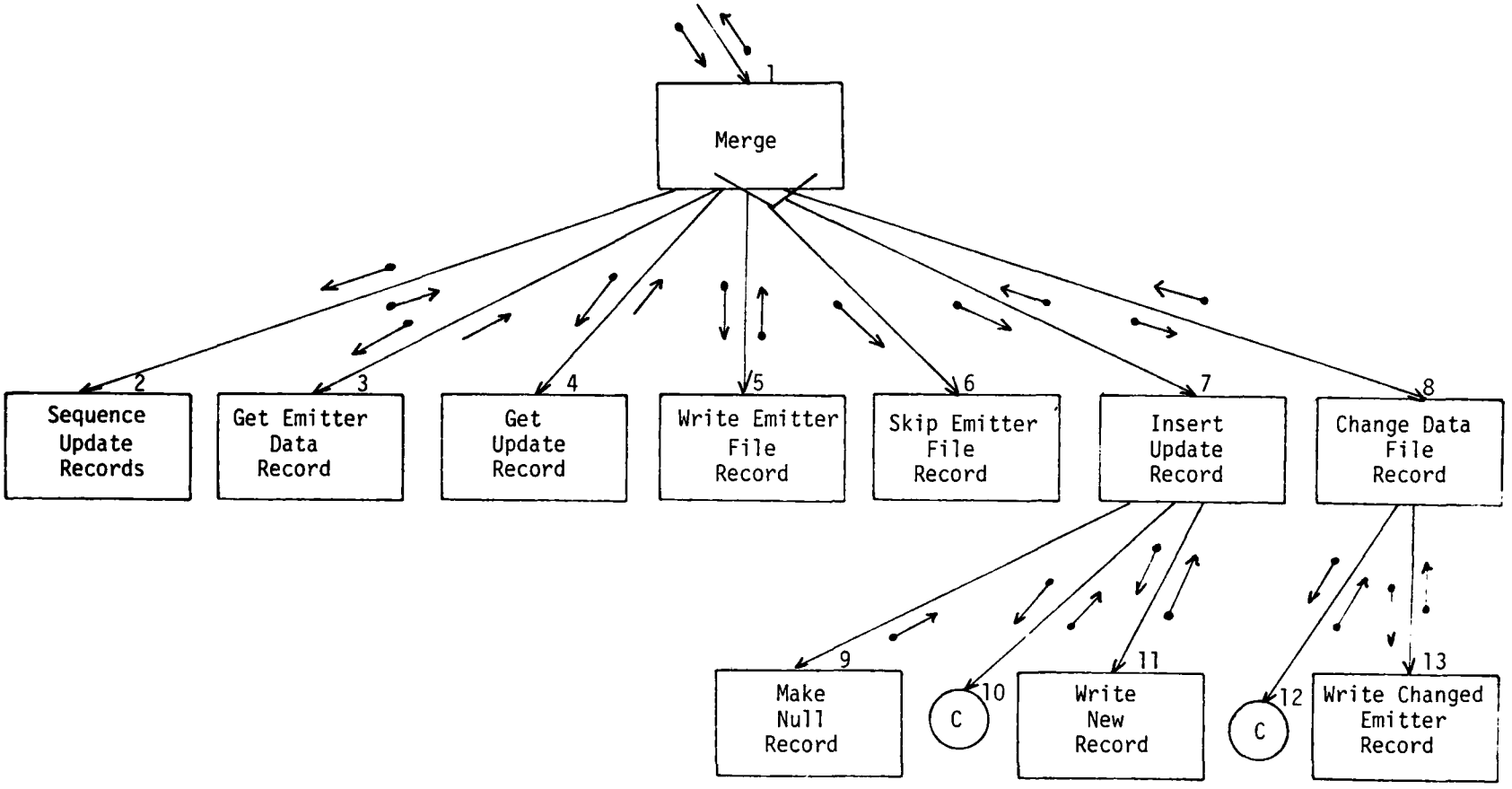



Figure 17. Merge

TABLE X

Parameters for Merge

| Input | Output |
|---|---|
| 1. Update Records, Emitter Data File | Updated Emitter Data File, Error Information |
| 2. Update Record Tree | Update Record Chain |
| 3. Emitter Data File | Emitter Data Record, End-of-Subfunction |
| 4. Update Record Chain | Update Record, End-of-Update Chain |
| 5. Emitter Data Record, New Emitter Data File | New Emitter Data File |
| 6. Emitter Data Record | - - - - - |
| 7. Update Record, New Emitter Data File | New Emitter Data File |
| 8. Update Record, Emitter Data Record New Emitter Data File | New Emitter Data File |
| 9. - - - - - | Null Emitter Record |
| 10. Update Record, Null Emitter Record | New Emitter Record, Error Information |
| 11. New Emitter Record, New Emitter Data File | New Emitter Data File |
| 12. Update Record, Emitter Data Record | Updated Emitter Data Record, Error Information |
| 13. Updated Emitter Data Record | New Emitter Data File |

## Module Description for the Merge Branch

Merge. This module takes the update records in the update record tree and merges them with the emitter data file. It performs the proper insertions, changes, and deletions. It controls the reading of the records and selects which of the modules to call to perform the update function that is requested.

Sequence Update Records. An inorder traversal of the update record tree would produce a sequential list of the update records. However, (as discussed in Chapter V) it was decided that performing this task once, instead of extracting records one at a time, would facilitate the merge process. This module performs the task of converting the binary tree structure to a linked change of records.



Get Emitter Data Record. This module retrieves the next record from the master emitter data file. It returns an EOF indicator when the master file has been exhausted.

Get Update Record. This module retrieves the next update record from the chain. It returns an end-of-chain indicator when all update records have been processed.

Write Emitter File Record. All records prior to the position for an insertion or a record to be deleted or modified are copied onto the new emitter data file. This module performs that task.

Skip Emitter File Record. This module performs the record deletion function. When an emitter

file record is found which matches an update record tagged for deletion, it is skipped. The next emitter data record and update record are fetched.

Insert Update Record. The insertion of a new emitter record is controlled by this module. When the position for the insertion is reached, the new record is built from the data in the update record. The new record is created in this module.

Make Null Record. This module clears the new record. Fields are set to 0, blank, or null values depending on the defined data type.

(C) Process Update, a module described as the next branch.



Write New Record. This module writes the new record out to the new emitter data file and returns to the merge control module to read the next update record.

Change Data File Record. The modification of an emitter data file record is accomplished by this module. It controls the modification of the emitter record that has been read from the current file and it is output to the new file.

(C) Process Update Record; a module described as the next branch.

Write Changed Emitter Record. This module writes the modified record out to the new emitter data file. Control is then passed back to the control module to read new update and emitter data records.

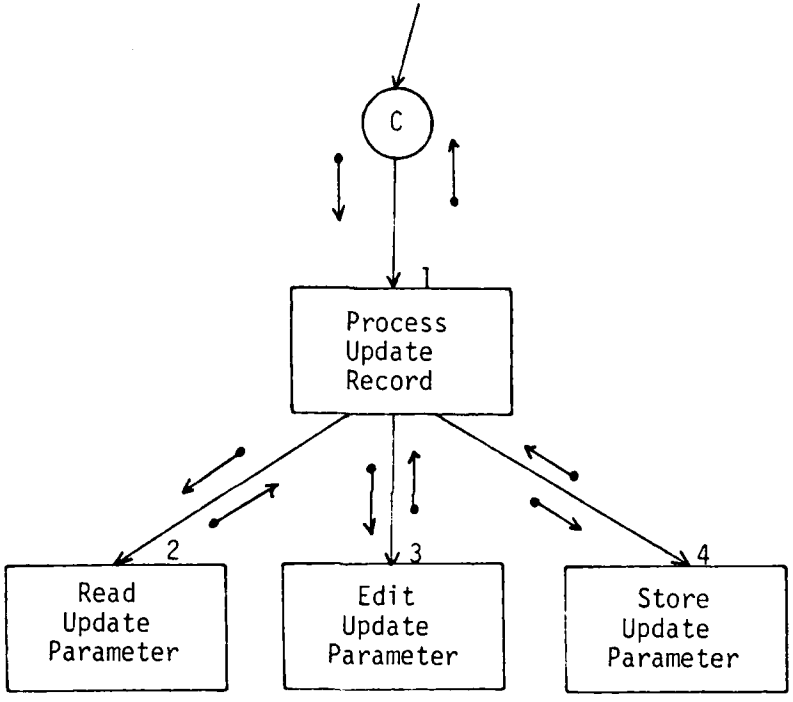



Figure 18. Process Update Record

TABLE XI

Parameters for Process Update Record

| Input | Output |
|---|---|
| 1. Update Record, Emitter Record | Updated Emitter Record, Error Information |
| 2. Update Record | Update Parameter |
| 3. Update Parameter | Edited Update Parameter, Error Information |
| 4. Edited Update Parameter, Emitter Record | Updated Emitter Record |

Called by: Insert Update Record, Change Update Record

## Module Description for the Process Update Record Branch

Process Update Record. The building and modifying of emitter data records is controlled by this module. It reads each chained parameter, edits it, and stores it in the new record.

Read Update Parameter. This module fetches the next parameter chained to the update record.

Edit Update Parameter. The parameters are edited to ensure that they match the fields indicated for their storage. Error information is returned to the control module. Overflow indicators are also returned.

Store Update Parameter. The parameter in the update record is stored in the emitter data record. If this is a change operation, it will overload old information, if it was available. An insert operation will load over a null value leaving fields, for which no data is available, with null values.

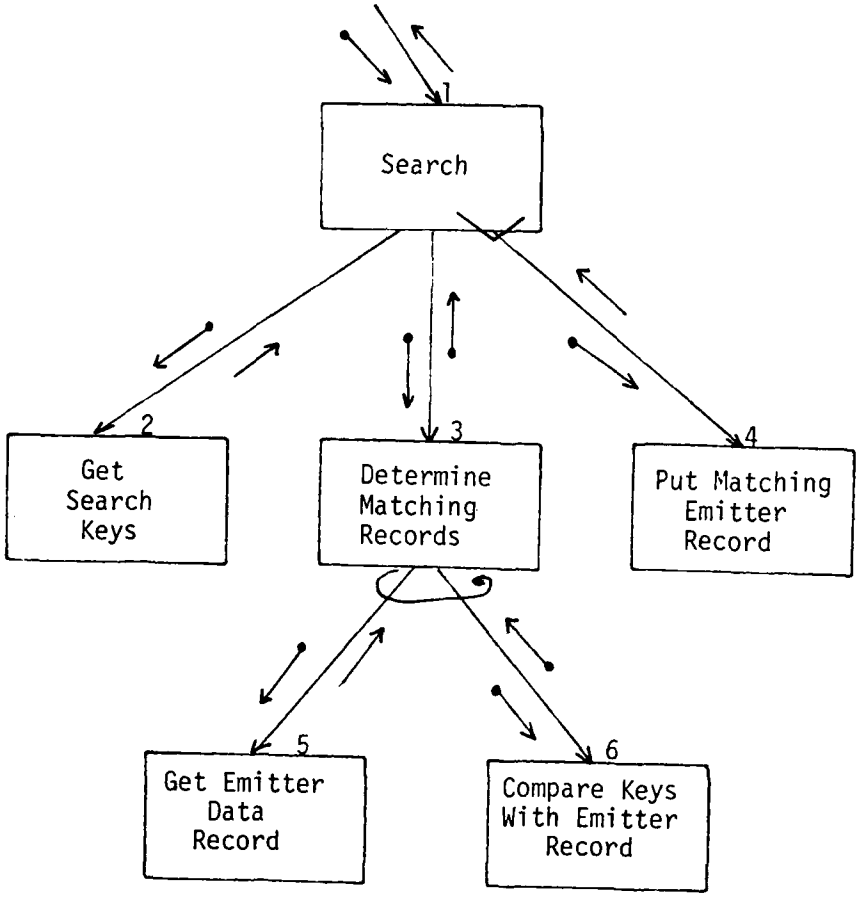



Figure 19. Search

TABLE XII

Parameters for Search

| Input | Output |
|---|---|
| 1. Search Control Cards, Emitter Data File | Error Information, Search Results, End-of-Function |
| 2. Search Control Cards | Search Keys, End-of-Search Keys, Error Information |
| 3. Search Keys Emitter Data File | Matching Emitter Numbers |
| 4. Emitter Data File, Search Control Cards, Matching Emitter Numbers | Search Results, End-of-Function, Error Information |
| 5. Emitter Data File | Emitter Data Record, End-of-Data File |
| 6. Search Keys, Emitter Data Record | Matching Emitter Numbers |

## Module Description for the Search Branch

Search. This is the control module for the system function, search. It monitors the input of the search keys, the retrieval of matching emitter data records, and the output of the search process. It also returns error and information messages to the user.

Get Search Keys. This module reads the search parameters from the input cards. It edits the keys for legality and stores legal keys in a linked list structure which will be used for the search process. Several sets of keys may be input to the search process. Error indicators are returned to the user. Data on legal keys is available in Appendix C.



Determine Matching Records. The actual search process is controlled by this module. It oversees the comparison of each emitter record with each set of search keys. The numbers of qualifying emitters are saved for later output.

Get Emitter Data Record. This module reads the next record from the master emitter data file. If no more files are present an EOF indicator is returned.

Compare Keys With Emitter Record. Each emitter record is compared with each set of keys by this module. For each set of keys that an emitter record matches completely, an entry is made, in a corresponding "match" list, of its emitter number.

Put Matching Emitter Records. This is the output routine for the search function. It determines the proper output format and writes the requested information for the user.

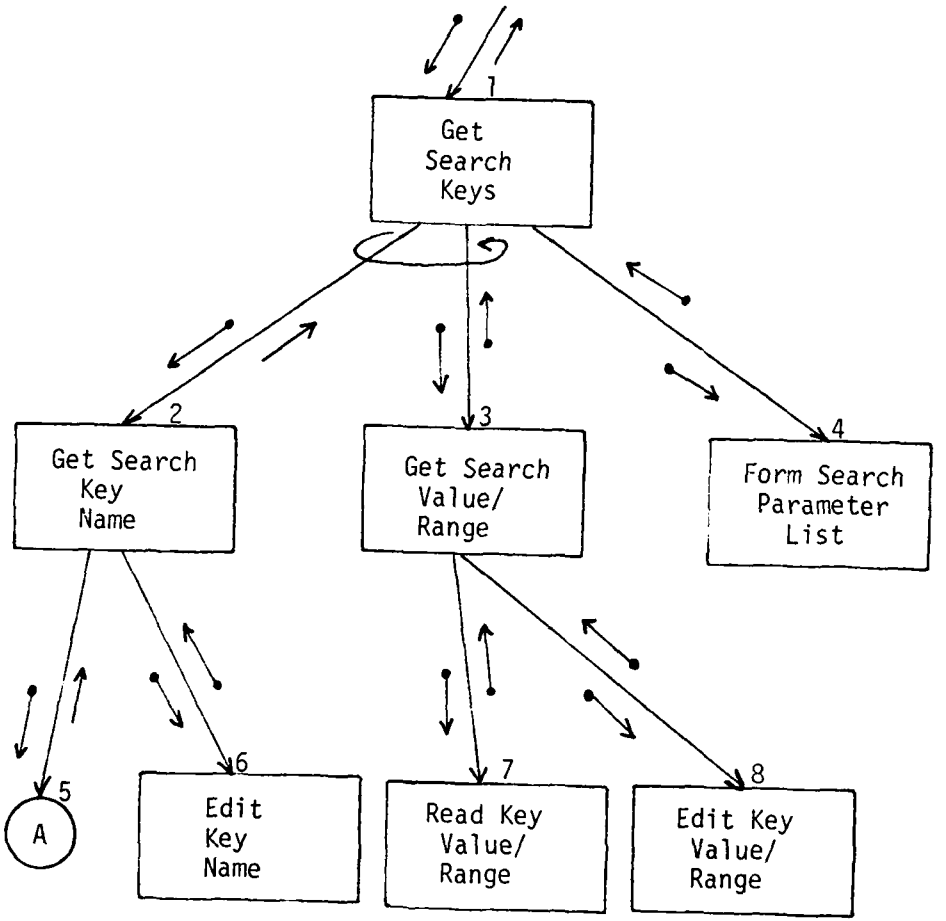



Figure 20. Get Search Keys

TABLE XIII

Parameters for Get Search Keys

| Input | Output |
|---|---|
| 1. Search Control Cards | Search Keys, End-of-Search Keys, Error Information |
| 2. Search Control Cards | Search Keys Names, End-of-Search Keys, Error Information |
| 3. Search Control Cards | Search Key Values/Ranges, Error Information |
| 4. Search Keys Names, Search Key Values/ Ranges | Search Keys |
| 5. Search Control Card | Names, End-of-Search Keys |
| 6. Name | Legal Search Key Name, Error Information |
| 7. Search Control Cards | Values/Ranges |
| 8. Values/Ranges | Search Key Values/Ranges, Error Information |

## Module Description for the Get Search Keys Branch

Get Search Keys. This module controls the user's input to the search process. Details on the structure involved is available in Chapter IV.

Get Search Key Name. The next input from the user is read and edited to determine if it is a legal search key. Error indicators are returned to the user.

(A) Get Card, a module described previously as a branch.

Edit Key Name. The control word, which was fetched, is edited to determine whether it is a legal search key (see Appendix C). It also returns an indicator to show the end of a set of keys and the end of all keys.

Get Key Value/Range. The value or range associated with each key is read and edited by this module. Error indicators are returned to the user.

Read Key Value/Range. This is the input module which obtains the value from the user's input cards.

Edit Key Value/Range. The value/range which was read is edited to determine whether it is the proper type for the associated key. Error indicators are returned to the user.

Form Search Parameter List. The legal key parameters are stored in a search data structure by this module. It keeps the sets of keys separate and sets up the structure in a manner which facilitates the search.

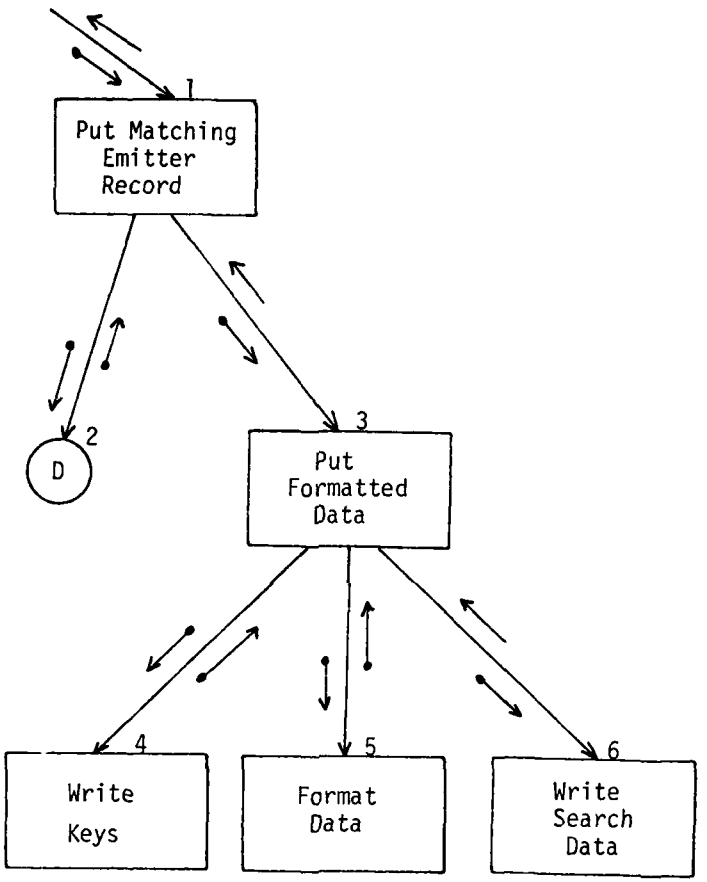



Figure 21. Put Matching Emitter Record

TABLE XIV

Parameters for Put Matching Emitter Record

| Input | Output |
|---|---|
| 1. Search Control Cards, Matching Emitter Records | Search Results, Error Information, End-of-Function |
| 2. Search Control Cards | Output List, Error Information |
| 3. Output List, Matching Emitter Records, Search Keys | Search Results, End-of-Function |
| 4. Search Keys | Search Results Heading |
| 5. Output List, Matching Emitter Records | Formatted Search Results |
| 6. Search Results Heading, Formatted Search Results | Search Results, End-of-Function |

## Module Description for the Put Matching Emitters Branch

Put Matching Emitters. This is the control module for the output of the search function. It monitors the format selected by the user and the output of the emitter data, from qualifying emitter data records, in that format.

(D) Get Output Format; a module described in the next branch.

Put Formatted Data. This module controls the actual writing of the emitter data in the requested format.

Write Keys. This module writes a header that contains the keys used for each query.

Format Data. The data from the emitter records found during the search process is formatted according to the results from the Get Output Format Branch. It puts out those characteristics that were chosen by the user, either explicity or by default.

Write Search Data. The data is printed for the user.

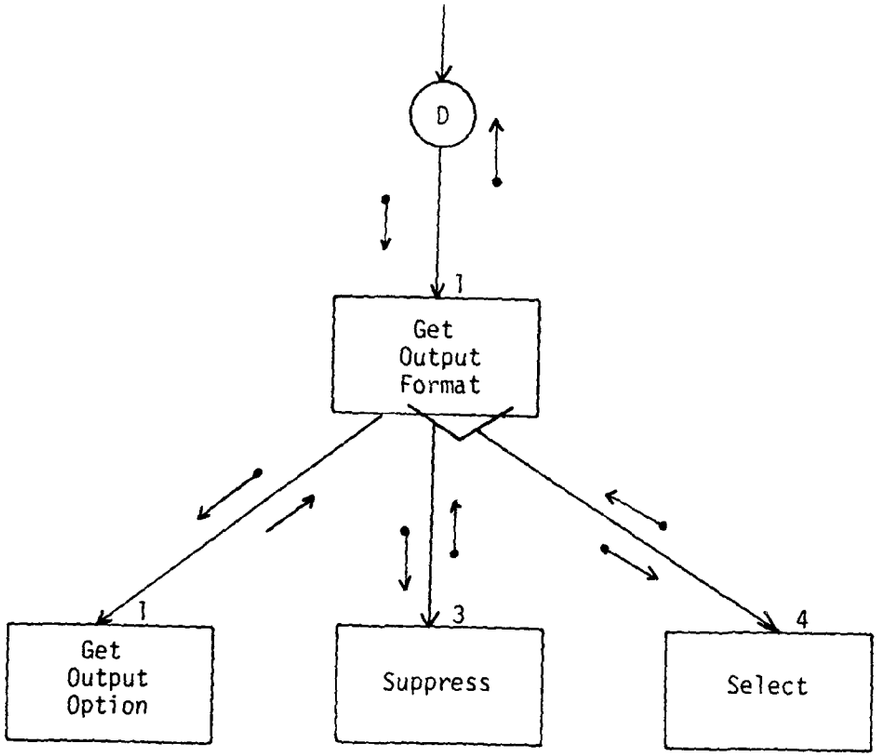



Figure 22. Get Output Format

TABLE XV

Parameters for Get Output Format

| Input | Output |
|---|---|
| 1. Output Control Cards | Output List, Error Information |
| 2. Output Control Cards | Output Option, Error Information |
| 3. Output Control Cards | Output List, Error Information |
| 4. Output Control Cards | Output List, Error Information |

Called by: Put Matching Emitters, Produce

## Module Description for the Get Output Format Branch

Get Output Format. The user may select or suppress the characteristics he wishes to see, on those emitters which passed the search tests. This module controls that process. If no option is specified, the default is a list of all characteristics.

Get Output Option. This module reads the next card to determine whether the select or suppress option is to be used. If no option is specified, control passes to the suppress module with no fields to suppress.

Suppress. A chain containing all output types is formed by this module. If the suppress option was specified, cards are then read and edited. All output types matching these cards are removed from the list. If no option was specified, the entire list is returned.



Select. This module forms an output list of all legal output types that are found on the control cards from the user.

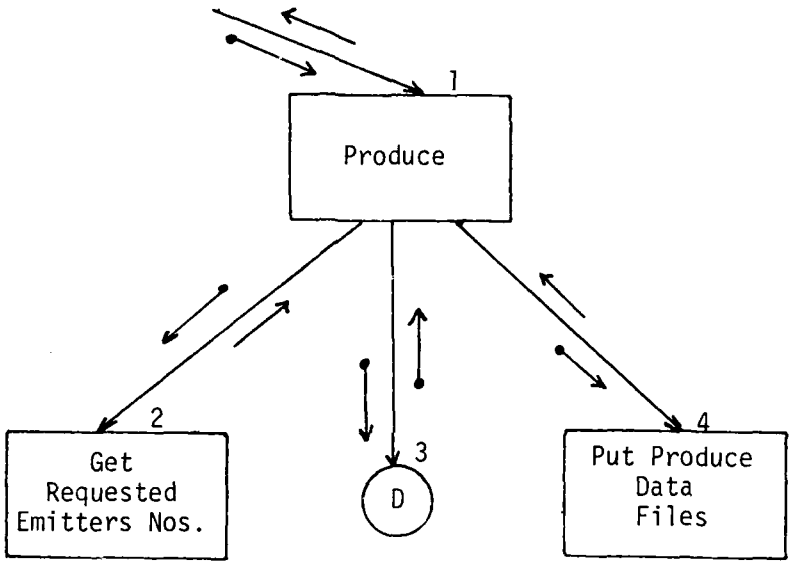



Figure 23. Produce

TABLE XVI

Parameters for Produce

| Input | Output |
|---|---|
| 1. Produce Control Cards, Emitter Data File | Problem Tape, Error Information, End-of-Function |
| 2. Produce Control Cards | Emitter Numbers, End-of-Input |
| 3. Produce Control Cards | Output List, Error Information |
| 4. Output List, Emitter Data File, Emitter Numbers | Problem Tape, Error Information, End-of-Function |

## Module Description for the Produce Branch

Produce. This is the control module for the Produce system function. It retrieves emitter records, based on emitter numbers supplied by the user, and writes the characteristics requested by the user to a tape, in a format compatible with the air defense models. Error and information messages are returned to the user.

Get Requested Emitter Numbers. This module reads the emitter numbers chosen by the user and stores those numbers in a tree structure which will be used to retrieve the requested records, then sequences the tree structure into a chain.



(D) Get Output Format; a module previously described as a branch.

Put Produce Data Files. This module retrieves the emitter data records, according to the numbers stored in the input chain, and writes a tape containing the characteristics stored on the output list.

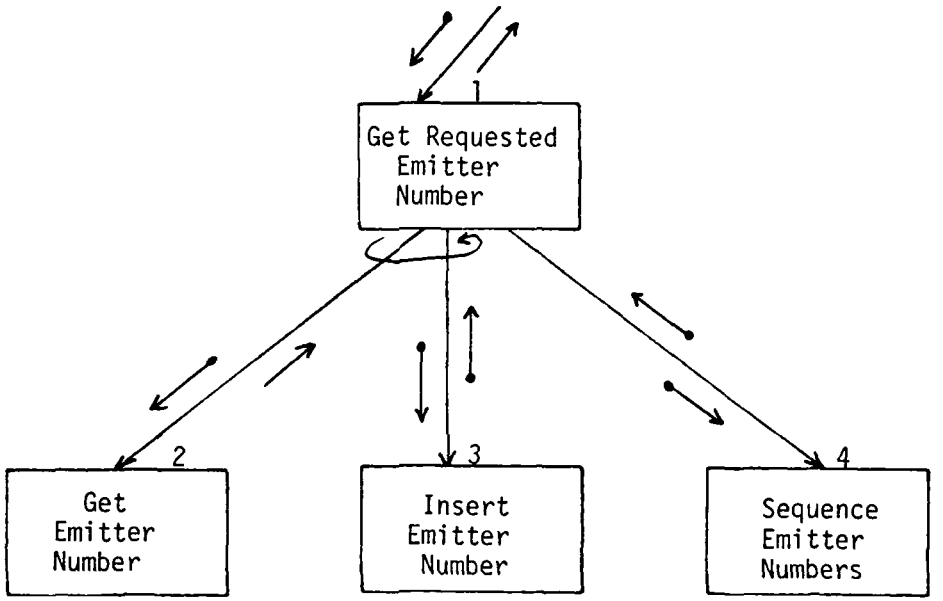



Figure 24. Get Requested Emitter Numbers

TABLE XVII

Parameters for Get Requested Emitter Numbers

| Input | Output |
|---|---|
| 1. Produce Control Cards | Emitter Number Chain, End-of-Input |
| 2. Produce Control Cards | Emitter Numbers, End-of-Input |
| 3. Emitter Numbers | Emitter Number Tree |
| 4. Emitter Number Tree | Emitter Number Chain |

## Module Description for the Get Requested Emitter Numbers Branch

Get Requested Emitter Numbers. This module controls the input to the produce function. It reads the requested emitter numbers and stores them for later processing.

Get Emitter Number. This module reads the next emitter number from the next card. It also returns an indicator when no emitter numbers remain.

Insert Emitter Number. The emitter number is stored in a tree structure which will facilitate the ordering of the numbers for retrieval.

Sequence Emitter Numbers. As described in Chapter V, the tree structure in this system were transformed into chains to facilitate processing. This module transforms the emitter number tree into an emitter number chain.

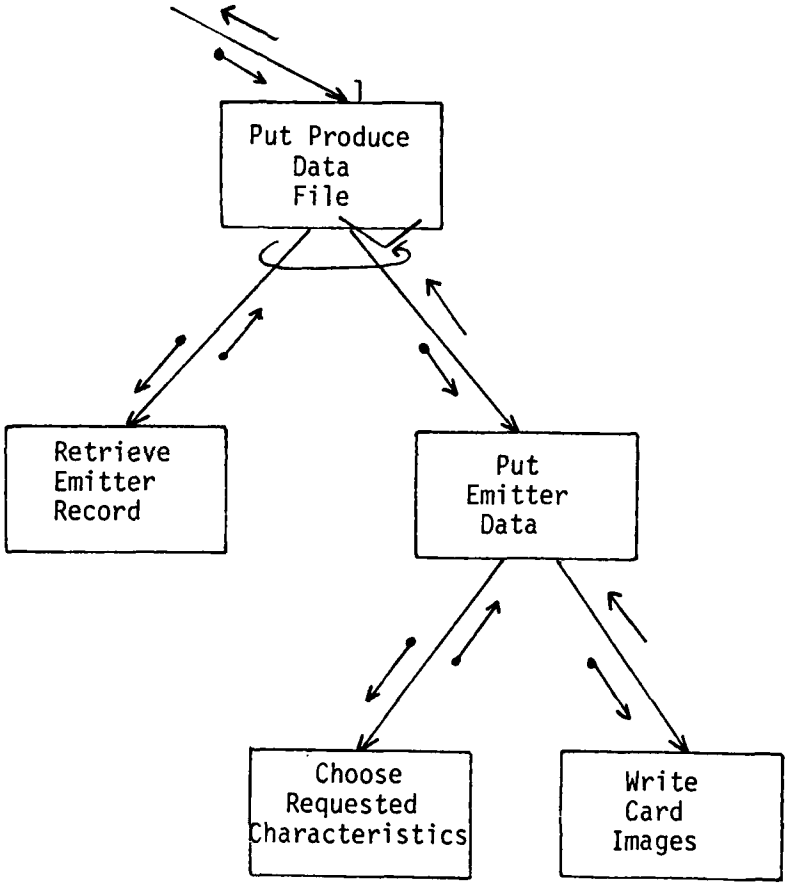



Figure 25. Put Produce Data File

TABLE XVIII

Parameters for Put Produce Data File

| Input | Output |
|---|---|
| 1. Output List, Emitter Data File, Emitter Numbers | Problem Tape, Error Information, End-of-Function |
| 2. Emitter Data File, Emitter Numbers | Emitter Data Records, Error Information |
| 3. Output List, Emitter Data Records | Problem Tape, End-of-Function |
| 4. Output List, Emitter Data Records | Formatted Problem Card Images |
| 5. Formatted Problem Card Images | Problem Tape, End-of-Function |

## Module Description for Put Produce Data File Branch

Put Produce Data File. The problem tape which contains electronic emitter data for the air defense models is produced by this branch. It retrieves the requested record and writes the chosen characteristics on the tape.

Retrieve Emitter Record. This module reads the master emitter data file until it finds the next requested record. This record is extracted for further processing. If the record is not found, an indicator is returned to the user.

Put Emitter Data. The retrieved record is written to the tape by this module. It uses the output list to select the proper characteristics, then prints those parameters in a card image format.



Choose Requested Characteristics. The output type list is used to select the characteristics that the user has chosen.

Write Card Images. The extracted characteristics are formatted in a fixed card image form to fit the input function of the air defense models. They are then written onto the tape.

Summary

These diagrams, tables, and module descriptions provide a modular breakdown for the system that has been created. They should provide the maintenance programmer with important information on the construction of the program. Further information is available in later chapters.

The Structured Design technique was a valuable tool. It greatly facilitated the designing of the system and the factoring of the requirements into small, easily programmable modules. The diagrams produced during this process should provide greater clarity into the System Design.

# IV. System Data Structures

## Introduction

The organization of the data, handled by this system, is the topic of this chapter. Specific definitions and data types of the data elements referred to in this chapter can be found in Appendix C and the references cited herein. This chapter, together with the two following constitute a programming guide for future maintenance programmers of this system. As stated in the title of a book by N. Wirth (Ref 7):

Data Structures + Algorithms = Programs.

This chapter describes the first argument of that equation. Each of the following sections describes the data structures for a major task of the program. The first contains the global data structures, which contain the data common to all sections of the program. Each function (update, produce, search) has its own data structures and a section of this chapter discusses each.

## Data Structures

Global Data. The global data structures for this program consist of the record structure for the emitter data file and tables, used by all the functions to edit and convert input data.

Each element of the master emitter data file is a record with the format shown in Figure 26. Table XIX gives a brief explanation of each field name. Several of the fields are also records and refer to Figures

27 - 35 and Tables XX - XXVIII. This record within a record format is easy to access in PASCAL. Several of the fields (e.g., BEAMS) consist of an array of records. This is necessary since a variable number of beams may be associated with one emitter.

| | |
|---|---|
| EMTRNUM | |
| SYSNAME | |
| RADARNAM | |
| TRANSMITR | |
| RECEIVER | |
| ELINTNUM | |
| MAJORSYS | |
| INDEX | |
| ECACNUM | |
| NFUNC | |
| FUNCKODE | |
| BEAMS | BEAMKOUNT |
| PRF | PRFKOUNT |
| IPPS | IPPSSRCE |
| PW | PWKOUNT |
| RF | RFKOUNT |
| SCANRATE | SCANKOUNT |
| NODRATE | NODKOUNT |
| RTRACK | RTRKSRCE |
| TTRACK | TTRKSRCE |
| WTRACK | WTRKSRCE |
| XTRACK | XTRKSRCE |
| CLASSIF | |
| EMTRCOMMNT | |

Figure 26. Emitter Data File Record

TABLE XIX

Fields of the Emitter Data File Record

| Field Name | Data Item Name | Data Type | Explanation/Comments |
|---|---|---|---|
| EMTRNUM | Emitter Number | Integer | Unique, system identifier |
| SYSNAME | System Name | Alpha | Full system name (20 Char) |
| RADARNAM | Short Name | Alpha | Abbreviated Name (10 Char) See Appendix C |
| TRANSMITR | Transmitter | Record* | See Table XXI, and Figure 28 |
| RECEIVER | Receiver | Record* | See Figure 29 and Table XXII |
| ELINTNUM | Elint Number | Alpha | Electronic Intelligence System Identifier |
| MAJORSYS | Major System Name | Alpha | The system of which this emitter is part |
| INDEX | | Integer | A number from 1-420 for model use |
| ECACNUM | ECAC System Number | Alpha | Electronic Capability Analysis Center |
| NFUNC | | Integer | System Parameter for Pulse Density Model |
| FUNCKODE | Function Code | Alpha | ECAC System Parameter |
| BEAMKOUNT | | Integer | Number of beams associated with this emitter |
| BEAMS | | Record* | See Figure 32 and Table XXV |
| PRFKOUNT | | Integer | Number of PRF values for this emitter |
| PRF | Pulse Repetition Freq | Record* | See Figure 30 and Table XXIII |
| IPPS | | Integer | Average PRF Value |
| IPPSSRCE | IPPS Source | Record* | See Figure 27 and Table XX |

AD-A080 416 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 5/2
AN EMITTER DATA MAINTENANCE AND RETRIEVAL SYSTEM.(U)
DEC 79 C C GARDNER
UNCLASSIFIED AFIT/GCS/EE/79-6 NL

2 OF 2
AD
A080416

END
DATE FILMED
3-80
DDC

TABLE XIX (Con't)

| Field Name | Data Item Name | Data Type | Explanation/Comments |
|---|---|---|---|
| PWKOUNT | | Integer | Number of PW values for this emitter |
| PW | Pulse Width | Record* | See Figure 31 and Table XXIV |
| RFKOUNT | | Integer | Number of RF values for this emitter |
| RF | Radio Frequency | Record* | See Figure 34 and Table XXVII |
| SCANKOUNT | | Integer | Number of Scan values for this emitter |
| SCANRATE | | Record* | See Figure 35 and Table XXVIII |
| NODKOUNT | | Integer | Number of Nod values for (Height Finder) |
| NODRATE | | Record* | See Figure 33 and Table XXVI |
| RTRACK | Max Detection Range | Real | |
| RTRKSRCE | RTRACK Source | Record | See Figure 27 and Table XX |
| TTRACK | Max Target Track | Real | Maximum Range for Target Tracking |
| TTRKSRCE | TTRACK Source | Record | See Figure 27 and Table XX |
| WTRACK | Max Lethal Range | Real | |
| WTRKSRCE | WTRACK Source | Record | See Figure 27 and Table XX |
| XTRACK | Min Lethal Range | Real | (Dead Zone) |
| XTRKSRCE | XTRACK Source | Record | See Figure 27 and Table XX |
| CLASSIF | Classification | Alpha | Overall Classification for emitter data |

TABLE XIX (Con't)

| Field Name | Data Item Name | Data Type | Explanation/Comments |
|---|---|---|---|
| EMTRCOMMNT | Emmiter Comments | Alpha | General comments on this emitter |
| | | *Array of Records | |

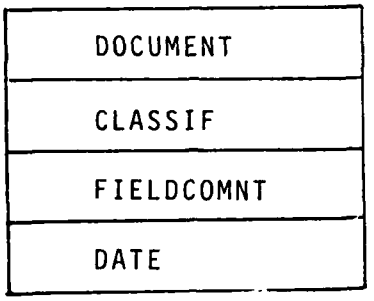



Figure 27. Source Record

TABLE XX

Fields for Source Record

| Field Name | Data Item Name | Data Type | Explanation/Comments |
|---|---|---|---|
| DOCUMENT | Source Document | Alpha | Source of the data with this record |
| CLASSIF | Classification | Alpha | Classification of data field |
| FIELDCOMNT | Field Comments | Alpha | Additional Comments on this data field |
| DATE | Date of Last Update | Alpha | Date of last update for this data |

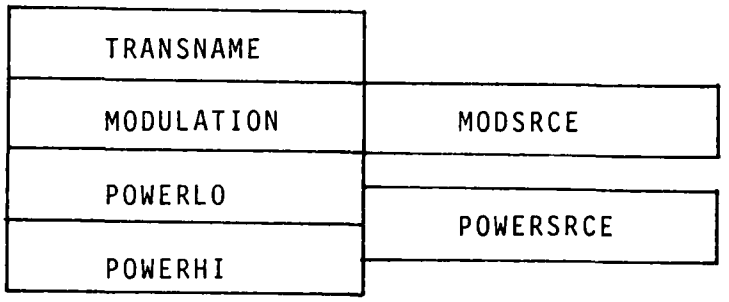



Figure 28. Transmitter Record

TABLE XXI

Fields for Transmitter Record

| Field Name | Data Item Name | Data Type | Explanation/Comments |
|---|---|---|---|
| TRANSNAME | Transmitter Name | Alpha | |
| MODULATION | Modulator Code | Alpha | ECAC System Code - Transmitter's Modulator |
| MODSRCE | Modulator Code Source | Record | See Figure 27 and Table XX |
| POWERLO | Transmitter Power | Real | Peak Power (Min of range) |
| POWERHI | | Real | Peak Power (Max of range) |
| POWERSRCE | Power Data Source | Record | See Figure 27 and Table XX |

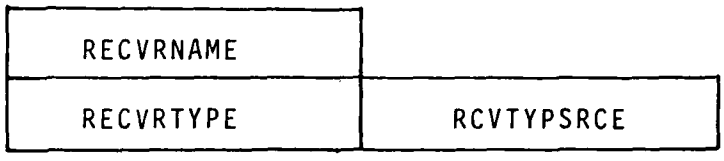



Figure 29. Receiver Record

TABLE XXII

Fields for Receiver Record

| Field Name | Data Item Name | Data Type | Explanation/Comments |
|---|---|---|---|
| RECVRNAME | Receiver Name | Alpha | |
| RECVRTYPE | Receiver Type | Alpha | |
| RCVTYPSRCE | Receiver Type Source | Record | See Figure 27 and Table XX |

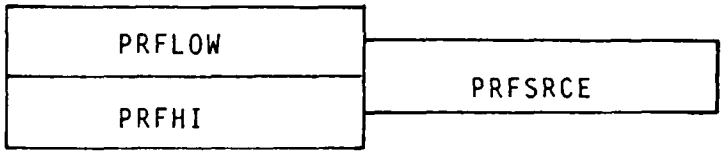



Figure 30. PRF Record

TABLE XXIII
Fields for PRF Record

| Field Name | Data Item Name | Data Type | Explanation/Comments |
|---|---|---|---|
| PRFLOW | PRF (Low) | Real | Low end of this PRF range |
| PRFHI | PRF (High) | Real | High end of this PRF range |
| PRFSRCE | PRF Source | Record | See Figure 27 and Table XX |

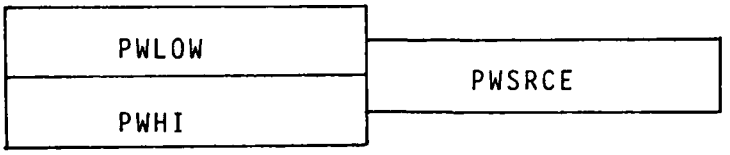



Figure 31. PW Record

TABLE XXIV

Fields for PW Record

| Field Name | Data Item Name | Data Type | Explanation/Comments |
|---|---|---|---|
| PWLOW | Pulse Width (Low) | Real | Low end of this PW range |
| PWHI | Pulse Width (High) | Real | High end of this PW range |
| PWSRCE | Pulse Width Source | Record | See Figure 27 and Table XX |

| ANTENNA | |
|---|---|
| IANT | |
| AGAIN | AGAINSRCE |
| HBW | HBWSRCE |
| VBW | VBWSRCE |
| TILTPHI | TLTPHISRCE |
| POLARIZE | POLARSRCE |
| BEAMFREQ | BMFRQSRCE |

Figure 32. Beam Record

TABLE XXV

Fields for Beam Record

| Field Name | Data Item Name | Data Type | Explanation/Comments |
|---|---|---|---|
| ANTENNA | Antenna Name | Alpha | |
| IANT | Antenna Pattern Type | Integer | Code (e.g., pencil, $CSC^2$ omni-directional etc) |
| AGAIN | Antenna Gain | Real | |
| AGAINSRCE | Antenna Gain Source | Record | See Figure 27 and Table XX |
| HBW | Horizontal Beam Width | Real | |
| HBWSRCE | HBW Source | Record | See Figure 27 and Table XX |
| VBW | Vertical Beam Width | Real | |
| VBWSRCE | VBW Source | Record | See Figure 27 and Table XX |

TABLE XXV (Con't)

| Field Name | Data Item Name | Data Type | Explanation/Comments |
|---|---|---|---|
| TILTPHI | Tilt Angle | Real | |
| TLTPHISRCE | Tilt Angle Source | Record | See Figure 27 and Table XX |
| POLARIZE | Polarization Code | Alpha | |
| POLARSRCE | Polarization Code Source | Record | See Figure 27 and Table XX |
| BEAMFREQ | Beam Frequency | Real | RF for this beam |
| BMFRQSRCE | Beam Frequency Source | Record | See Figure 27 and Table XX |

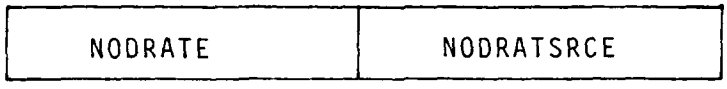



Figure 33. Nodrate Record

TABLE XXVI
Fields for Nodrate Record

| Field Name | Data Item Name | Data Type | Explanation/Comments |
|---|---|---|---|
| NODRATE | Time for One Nod | Real | Height Finder Data |
| NODRATSRCE | Nodrate Source | Record | See Figure 27 and Table XX |

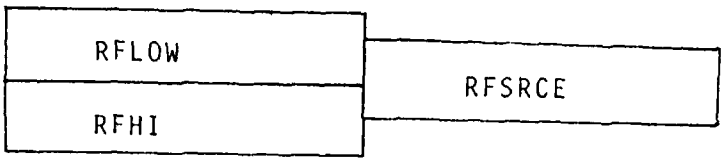



Figure 34. RF Record

TABLE XXVII
Fields for RF Record

| Field Name | Data Item Name | Data Type | Explanation/Comments |
|---|---|---|---|
| RFLOW | Radio Frequency (Low) | Real | Low end of this RF range |
| RFHI | Radio Frequency (High) | Real | High end of this RF range |
| RFSRCE | Radio Frequency Source | Record | See Figure 27 and Table XX |

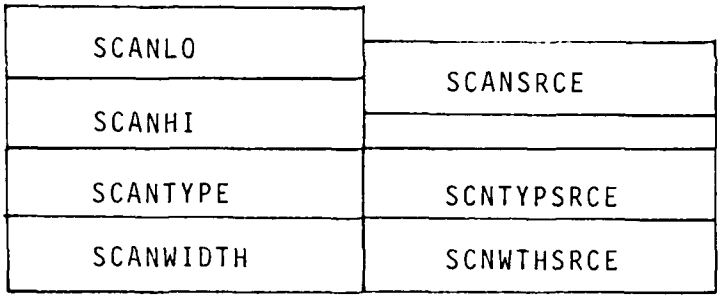



Figure 35. Scan Record

TABLE XXVIII

Fields for Scan Record

| Field Name | Data Item Name | Data Type | Explanation/Comments |
|---|---|---|---|
| SCANLO | Scanrate (Low) | Real | Low end of this Scanrate range |
| SCANHI | Scanrate (High) | Real | High end of this Scanrate range |
| SCANSRCE | Scanrate Source | Record | See Figure 27 and Table XX |
| SCANTYPE | Type of Scan | Alph | (e.g., circular, sector)-ECAC |
| SCNTYPSRCE | Scantype Source | Record | See Figure 27 and Table XX |
| SCANWIDTH | Scan Width | Real | |
| SCNWTHSRCE | Scan Width Source | Record | See Figure 27 and Table XX |

The specific contents of each field is not vital knowledge for the programmer, but the structure helps explain the accessing method used by each of the functions. Utilizing this structure simplified the input/output processes involving the master emitter data file. Each record is read/written as a unit, rather than character by character to a text file. A linked list of records for the variable size fields would save memory and avert the possibility of an overflow. However, memory use is not important in the environment in which this system will exist, and overflow will be an unusual and not a disastrous occurrence. Ths use of a record by record I/O saved many programming errors.

The other global record structure is the output type record (Figure 36). It is used by the Search and Produce Functions to monitor which characteristics have been selected, by the user, for output.

Two tables are used by the system to convert numbers to characters and characters to numbers. Each table consists of ten entries (Tables XXX-XXXI). One table is accessed as an array with entries numbered 0-9. Each entry contains the character code used by the machine for the corresponding digit ("0" - "9"). The other table reverses the process. The array is accessed by character code and the entries contain the digits 0-9. The PASCAL standard functions CHR and ORD could have been used for the conversions, but the tables minimized function calls.

Two other tables are used for editing purposes. One contains all legal card types that may be input to the update process and must be output during the produce function. The other table contains all legal output types. This table is used to edit the output types selected by the user for the Search and Produce Functions. There are 41 legal input and output types. Appendix C lists and explains each.

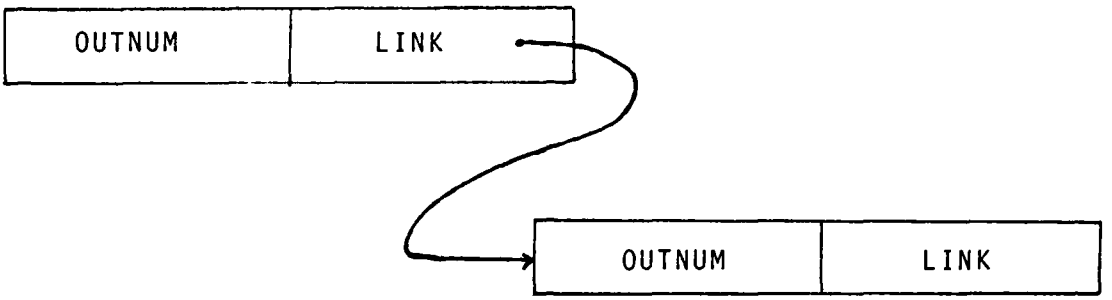



Figure 36. Output Type Record

TABLE XXIX

Fields for Output Type Record

| Field Name | Data Item Name | Data Type | Explanation/Comments |
|---|---|---|---|
| OUTNUM | Output Type | Integer | Code for an Output type See Appendix C |
| LINK | Next Output Type | Pointer | Points to next output type or is null if no more exist |

TABLE XXX

Number to Character Conversion Table

| Index | Entry |
|---|---|
| 0 | "0" |
| 1 | "1" |
| 2 | "2" |
| 3 | "3" |
| 4 | "4" |
| 5 | "5" |
| 6 | "6" |
| 7 | "7" |
| 8 | "8" |
| 9 | "9" |

TABLE XXXI

Character to Number Conversion Table

| Index | Entry |
|---|---|
| "0" | 0 |
| "1" | 1 |
| "2" | 2 |
| "3" | 3 |
| "4" | 4 |
| "5" | 5 |
| "6" | 6 |
| "7" | 7 |
| "8" | 8 |
| "9" | 9 |

Update Function Data. The update function has three data structures of its own. They are used to store and process data concerning modifications to the emitter data file. These structures are local to the update section of the program and are disposed of prior to an exit from that section.

The update record (Figure 37) is used as a node in an update tree. The data, input by the user, is used to sequence the inputs by emitter number. Each node is tagged with an update function (delete, change, or insert) and can point to two other update records. Storage and access of this tree is discussed in Chapter V.

The modification records (Figure 38) are chained to the update record. They contain the actual data to be chained or inserted. (A delete record has no modification record attached.)

Once the update record tree has been built, it is processed into a chain. The chain records (Figure 39) are used for this purpose. Each has a node from the update tree attached and is in turn, attached to the next record in sequence.

Search Function Data. Several data structures are used in the Search Function to enable the searching to be performed in one pass through the emitter data file. The organizations chosen facilitated this task.

The query master record (Figure 40) is the key data structure for this process. The parameters to be used for a search are chained to the master record as they are received. During the processing, matching emitter numbers are chained to the master record as qualifying emitter data records are discovered. The actual search algorithm is discussed in Chapter V.

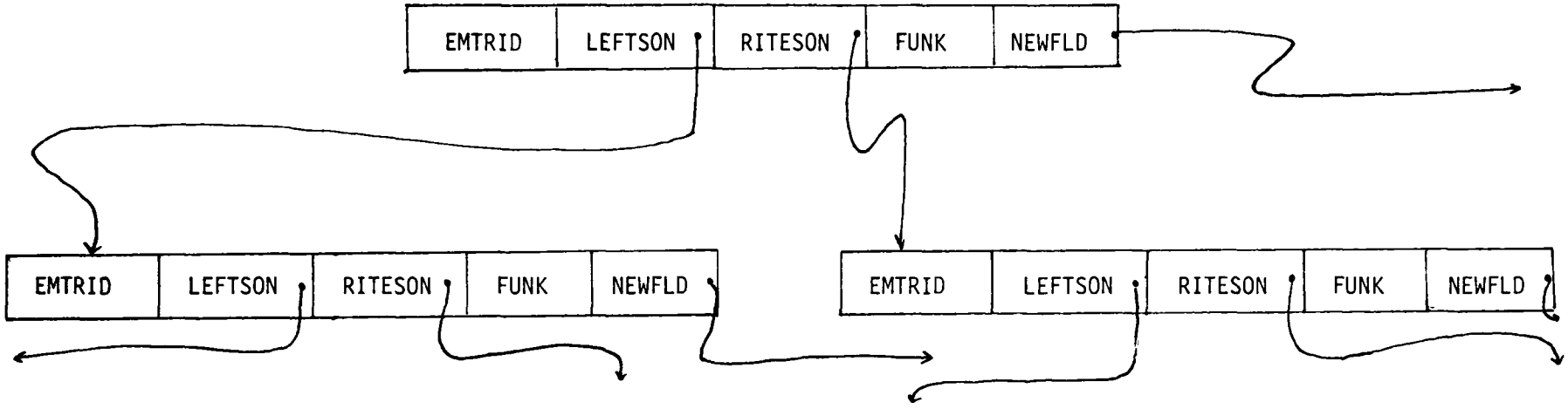



Figure 37. Update Record



TABLE XXXII

Fields for Update Record

| Field Name | Data Item Name | Data Type | Explanation/Comments |
|---|---|---|---|
| EMTRID | Emitter Number | Integer | |
| LEFTSON | | Pointer | Points to next record added with emitter number less than this record |
| RITESON | | Pointer | Points to next record added with higher emitter number |
| FUNK | Update Function | User-Defined | Delete, Change, or Insert |
| NEWFLD | Update Parameter Chain | Pointer | Points to first modification record (Figure 38) |

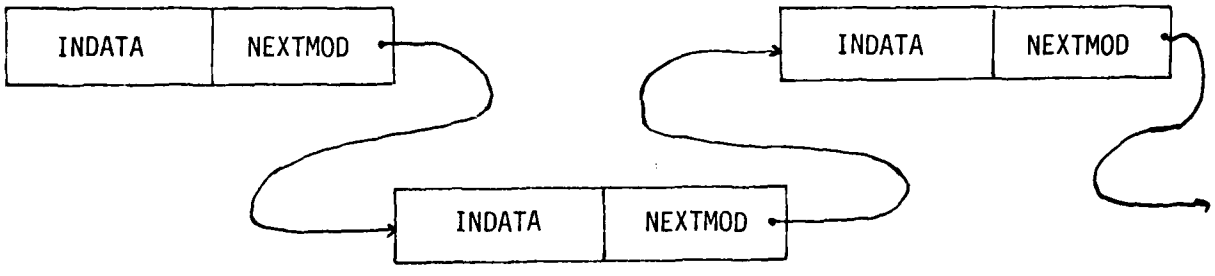



Figure 38. Modification Record

TABLE XXXIII

Fields for Modification Record

| Field Name | Data Item Name | Data Type | Explanation/Comments |
|---|---|---|---|
| INDATA | Modification Parameter | Alpha | Holds parameter from input card |
| NEXTMOD | Next Modification Record | Pointer | Points to next modification record for this emitter |

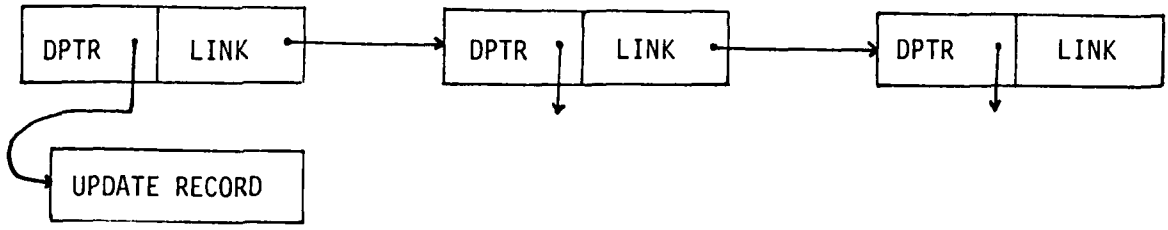



Figure 39. Chain Record

TABLE XXXIV

Fields for Chain Record

| Field Name | Data Item Name | Data Type | Explanation/Comments |
|---|---|---|---|
| DPTR | Update Data | Pointer | Points to an Update Record |
| LINK | Next Chain Record | Pointer | Points to the next chain record |

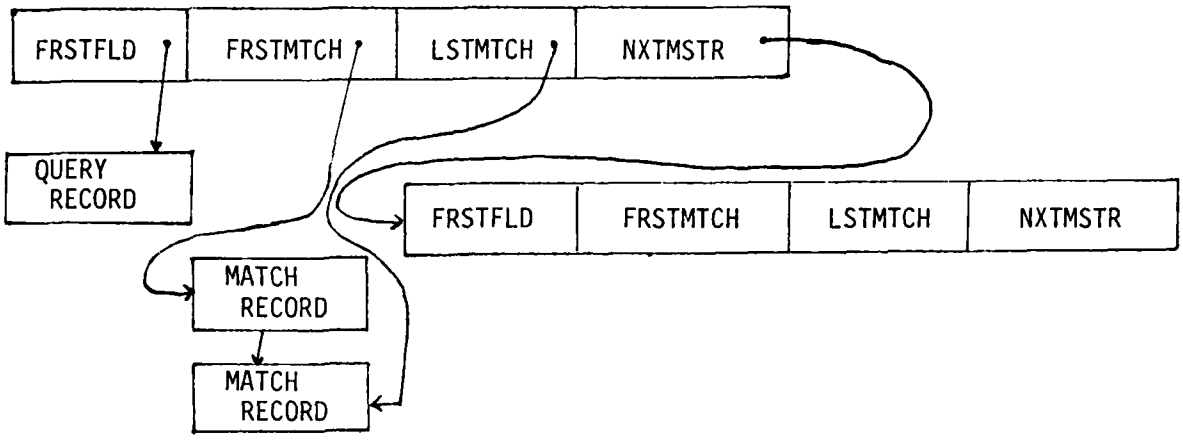



Figure 40. Query Master Record

TABLE XXXV

Fields for Query Master Record

| Field Name | Data Item Name | Data Type | Explanation/Comments |
|---|---|---|---|
| FRSTFLD | First Search Key | Pointer | Points to first Query Record |
| FRSTMTCH | First Matching Emitter | Pointer | Points to first Match Record |
| LSTMTCH | Last Matching Emitter | Pointer | Points to last match Record |
| NXTMSTR | Next Query Master | Pointer | Points to next Query Master Record |

The query record (Figure 41) contains the keys and values to be used for each search. These are chained to the master, as described above. The match records (Figure 42) contain the emitter numbers of all qualifying records.

Produce Function Data. The produce function also uses a tree structure. This is used to sequence the emitter numbers input by the user. A sequential list of emitter numbers allows problem tape to be produced in one pass through the emitter data file.

The nodes of the tree consist of a simple record structure (Figure 43). The emitter number and two pointers are the only fields in this record.

As discussed in the update section, trees such as this one are sequenced into chains to facilitate further processing. The chain record (Figure 44) is used for this purpose.

## Summary

The data structures described in this chapter are fundamental to the processing of the system. A knowledge of their makeup and use (Chapter V) enables the maintenance programmer to readily expand or convert this system. Further information on the basic data structures utilized is available in Wirth (Ref 7) and Ajo et al (Ref 1).

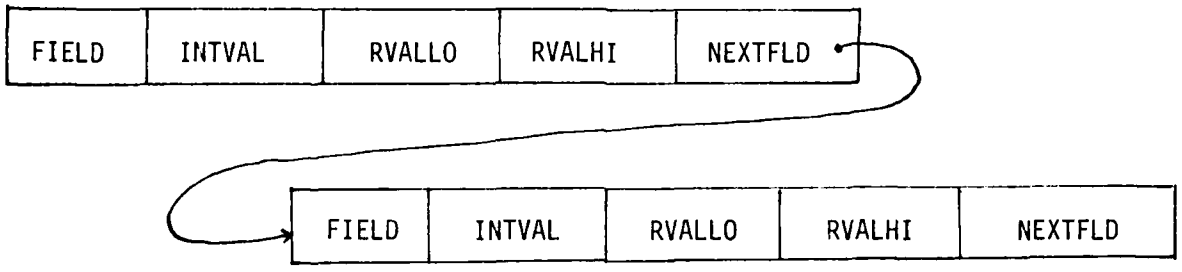



Figure 41. Query Record

TABLE XXXVI

Fields for Query Record

| Field Name | Data Item Name | Data Type | Explanation/Comments |
|---|---|---|---|
| FIELD | Search Key Name | Integer | Code for a Search Key (See Appendix C) |
| INTVAL | Search Key Value | Integer | Integer value for Search |
| RVALLO | Search Key Range | Real | Low end of Search Range |
| RVALHI | Search Key Range | Real | High end of Search Range |
| NEXTFLD | Next Search Parameter | Pointer | Points to the next Query Record |


Figure 42. Match Record

TABLE XXXVII

Fields for Match Record

| Field Name | Data Item Name | Data Type | Explanation/Comments |
|---|---|---|---|
| ENUMBER | Emitter Number | Integer | Emitter number of a record which matched this set of parameters |
| NXTMTCH | Next Match Record | Pointer | Points to the next matching emitter number |

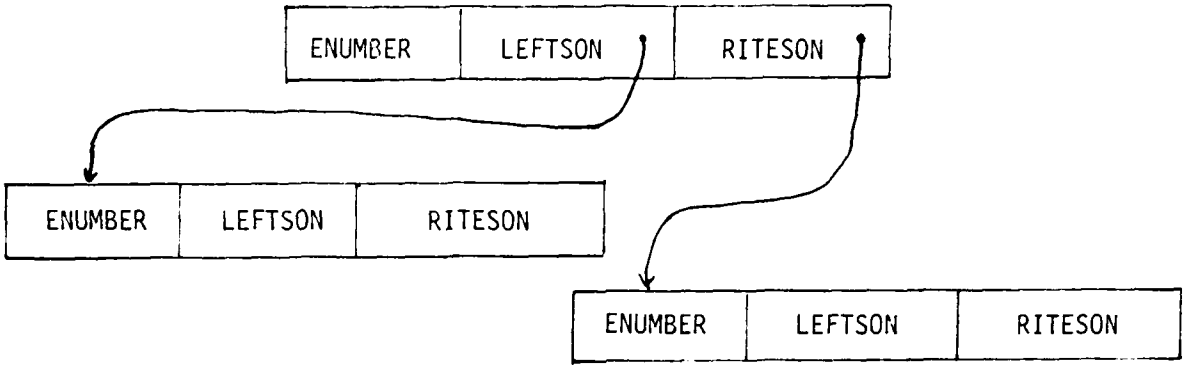



Figure 43. Emitter Number Record

TABLE XXXVIII

Fields for Emitter Number Record

| Field Name | Data Item Name | Data Type | Explanation/Comments |
|---|---|---|---|
| ENUMBER | Emitter Number | Integer | Number of the emitter to be processed |
| LEFTSON | | Pointer | Points to left child in binary tree structure |
| RITESON | | Pointer | Points to right child in binary tree structure |

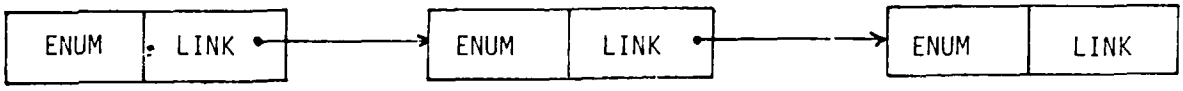



Figure 44. Chain Record

TABLE XXIX

Fields for Chain Record

| Field Name | Data Item Name | Data Type | Explanation/Comments |
|---|---|---|---|
| ENUM | Emitter Number | Integer | |
| LINK | Next Emitter | Pointer | Points to next emitter to be processed |

# V. System Algorithms

## Introduction

The data structures described in the previous chapter must be accessed and manipulated to produce the desired results. This chapter describes the major algorithms used to perform these tasks. It provides the maintenance programmer with an overall picture of how system functions are realized.

In general, algorithms were chosen because they best met the design constraints discussed in Chapter II. For example, a new search routine was written because information retrieval pieces of existing software packages would not be readily transportable from the CDC system to the new VAX 11/780 system proposed as the future environment for this system. Other choices based on design constraints are related in the appropriate section below.

## Algorithms

Executive Routine Operation. The executive module of the program is purposefully simple. It is mentioned here only to provide continuity into the system functions. It sets up the editing tables discussed in Chapter IV and calls the reading and editing modules as shown in Chapter VI.

Update Routine Operation. The update record tree was discussed in Chapter IV. Two methods of using this data structure are important to the operation of the update routine. The first method is used to insert a new update routine into the tree structure. This is a recursive routine

which searches a binary tree structure until it finds the proper position to insert the new record. The routine will replace previous change or insert records. The update function change, however, may need to add new information to a record that already exists in the tree (from a previous change or insert). It uses a routine to attach the new information to the proper node in the tree. This routine searches the binary tree structure also, but when a node, containing the proper emitter number is found, it then searches the chain of information associated with that node and attaches the new information at the end (unless the node was tagged as a delete).

When the update records are ordered sequentially by emitter number, they are merged into the existing emitter data file. This method, sort and merge, allowed a one pass update of a sequential file. No more than one master emitter data record need be in core at one time. Security considerations (see Chapter II) necessitated the use of tape storage for the emitter data file. This made the above updating algorithm very attractive.

The merge routine reads an update record, then reads the emitter data file. It copies each emitter data record onto the new file until one is reached with an emitter number equal to or greater than the update record. If the update record is a delete and the emitter numbers match, the emitter data record is skipped and new records are read; processing continues. Change records are used to modify the emitter records before they are copied to the new files. Insert records are used to build new records which are copies to the new file in the proper sequence. Error messages are produced when improper updates (e.g., deleting a nonexistent record) are attempted. When no update records remain, all remaining

emitter data records are copied to the new file. If the old file is exhausted first, the remaining insert records are processed.

Search Routine Operation. The general concept behind the search algorithm designed for this system is a one pass search which can compare each emitter data record against several sets of search parameters and store the emitter numbers of those that qualify against a set. Since processing time was not critical for this system, this is a clear, simple method which can be easily transported to another computer system. Since the design constraints forced a sequential storage medium, this was a reasonable solution.

As discussed in Chapter IV, the search parameters are chained to a master record. Each emitter data record can be examined against each set of parameters. Each element in the chain is compared with the characteristics present in the record being examined. If a failure occurs, the process moves to the next chain of parameters. If all tests are satisfactory, the number of that emitter data record is chained to the master record. Processing again moves to the next chain. When all chains have been processed, a new emitter record is read.

While one pass was sufficient to search the data file, a second pass is used to output the desired emitter characteristics. This is a simple straight forward routine. The select/suppress options are detailed in another section below.

Produce Routine Operation. The produce routines also use a tree structure (see Chapter IV). Again, a recursive routine is used to insert emitter numbers. Another recursive routine is used to reorganize the tree structure into a chain. The actual processing is a straight forward,

sequential search through the emitter data file extracting information requested by the user for output onto a problem tape.

Select/Suppress Routine Operation. These routines are considered separately because they are used by both the Search and the Produce functions. For these routines, Select implies that only the types explicitly requested will be output and Suppress means that all output types are used except those specifically deleted by the user. If a select control card is read, a chain of user selected output types is formed to be used in selecting emitter characteristics for output. If no option or the suppress control card is input, then a chain containing all types is formed. If the suppress option was requested, the types to be suppressed are read in and removed from the output chain.

## Summary

None of the algorithms used for this system is particularly complicated. Simplicity and ease of maintenance were chosen as being of higher priority than processing speed. Details on control and data flow for the routines are provided in Chapter III. Chapters IV and V are written to provide a summary of the methods used to implement this system. The next chapter will give detailed advice for use in converting or expanding this system.

# VI. Maintenance Programming Considerations

## Introduction

One factor which influenced several design decisions for this project was the forthcoming conversion of the program to enable it to run on a VAX 11/780 computer system. This chapter sets down the modifications that must be made or at least considered during this conversion process. It also contains a discussion of the areas to be examined while expanding the capability of a function or adding a function. While not necessarily complete, the information in the sections below provides the maintenance programmer with a useful checklist to aid the modification.

The first section concerns program conversion. Portability was a major goal in the system design, however, certain modifications will be necessary. Modification of the emitter data file is discussed in the next section. The user may desire to add characteristics to the emitter records or increase the size of the records. This can be accomplished with minimal programming effort. The last section concerns adding a function to the system or expanding one of the functions that already exist. Some of the possible enhancements are included in the recommendations for future modifications set forth in Chapter VII.

## Maintenance Considerations

Program Conversion. The first consideration which must be taken into account is the portability of the programming language. With one

exception, the PASCAL enhancements, known as PASCAL 6000-3.4 (see PASCAL User's Manual, Ref 5:88-103), were not employed in this system. These enhancements were added for the CDC computer system and may not be transportable to the VAX system. The exception is the ALFA data type. This is a ten character packed array. These ten characters can be packed into one computer word on the CDC computer. The VAX 11/780 can pack four characters per word. This will necessitate several large changes to the program.

The simplest method to solve this problem would be to define a data type ALFA as a four character packed array. The next step involves redefining the alphanumeric fields fo the emitter data record. For example, the transmitter name stored in the transmitter record can be up to 20 characters long. Currently the name is stored in two ten-character fields, TRANSNAME AND TRANSNM2. This would be expanded to five four-character fields for the VAX system; possibly TRANSNAME, TRANSNM2, TRANSNM3, TRANSNM4, TRANSNM5. Tables XIX-XXXIX list the alphanumeric fields that must be modified in this manner. In addition, data type CHARRAY is defined as a ten entry array of characters. This must be changed to a four entry array and the system constant MAXCOMMNT must be changed from 7 to 20.

The processing of these fields will also be affected. All system functions use these alphanumeric fields. Additional stores and fetches must be added to the processing portion of the functions to use the additional fields. Since no alphanumeric searches are performed, no comparison need be modified in the search section.

Alphanumeric comparisons are performed during the editing of control cards. Modifications to these tests must be made. The simplest method is to only compare the four leading characters of a control word. This would only require changes to the constants containing the legal values. For example, the constraints assigned, during the initialization procedure, to the output type table could be changed to four character constants. Care should be taken with two types, however, Output Type 22 is 'SCAN ' and Output Type 23 is 'SCAN TYPE'. The first four characters are identical. The user and maintenance programmer must decide whether to modify the user-created output types or to add the additional editing checks to check more than four characters.

Another area which is affected by this character per word situation is the temporary alphanumeric variable used in the program. Table XL lists, by procedure, those that will need attention either by redefinition or by modifications to their use.

Throughout the source code produced for this project, comments have been inserted indicating the areas requiring modification. These comments are general and not specific to the conversion to the VAX. They may remain after conversion to the VAX as flags to programmers required to perform future conversions.

The remainder of the language is standard PASCAL. No external procedures or other machine dependent enhancements were used. The algorithms and data structures used were also selected for their portability. The VAX system will be secure and storage of the emitter data file will be on disk rather than tape. The sequential processing used for these system functions should perform satisfactorily on this computer. Attention

should, however, be given to the operating system of the new computer. If it does not automatically place the new data file in permanent disk storage, a read/write mass storage routine might be necessary.

TABLE XL

Alphanumeric Work Variables Affected by Conversion

| Global | Update | Search | Produce |
|---|---|---|---|
| CWORD | SUBFUNC | QFLD | OUTTYPE |
| OUTTYPE | OUTFUNK | TEMP | OUTARRAY |
| | TEMPALFA | | TEMPARRAY |
| | MSGVAR | | |
| | TEMPARRAY | | |
| | TEMP | | |

Emitter Data File Modification. Two possible modifications can be envisioned for the emitter data file. New characteristics can be added to records or more instances of existing characteristics could be required. Neither of these require drastic program changes.

The addition of a new characteristic requires several simple changes. The first step is to add the definition of the necessary fields to the appropriate record. For example, a new receiver characteristic would be defined within the receiver record. If the characteristic has a source document of importance, a source field should also be defined. Next, modifications to the update function must performed. These will allow the field(s) to be loaded with the proper value. The Make Null procedure is modified to initially set the field(s) to a null value.

A new card type and output type must be added to the tables set in the initialization procedure and the MAXTYPE constant must be increased by 1. The produce function must be modified to output a new card image, if the characteristic is to be output to the air defense models. If the new characteristic is an integer or real value then it may be used as a search key. The search routine is modified to compare on that key. All new characteristics are added to the search output procedure.

The position of all these modifications within the system functions is determined by the position of the new characteristic inside the card type and output type tables. All input, output, and search processing is performed using a CASE statement. A new "Case" is added at the appropriate position.

The most difficult problem involved in adding characteristics is rebuilding the data file. After recompiling the program, a new file must

be built even if no values are available for the records. The simplest method involves performing a produce to retrieve a tape of card images for the entire file. This can be used as an input to the new program. Comments, however, must be added by hand.

Adding new instances of existing characteristics can be performed with one change. The constant, setting the size of the array of records holding that characteristic, is changed. For example, if additional beams are desired, MAXBEAM is increased. The loading and retrieval of additional beams is accomplished automatically.

System Expansion. A discussion of a system enhancement can only be accomplished in general terms. Major changes will require complete knowledge of the existing program. Even the recommendations discussed in Chapter VII are not simple.

The addition of a new function, involves changes to the executive portion of the program. The editing and interpretation modules must be modified to accept new control cards and branch to the new procedures. If portions of existing functions are to be used, they must be moved as modules, into a global area of the program.

Modifications to existing functions should be done carefully. Thorough testing of existing functions must be reaccomplished whenever a module is modified. The coupling between modules has been kept to a minimum but modifications to existing modules can have a ripple effect throughout the code. The design specifications should be examined to determine which modules interact with each other. Above all, all system changes should be accompanied by corresponding changes to the User's Manual (Appendix C) and the system design specifications (Chapter III). When the documentation becomes obsolete, the program will soon follow.

# VI. Conclusions and Recommendations

## Introduction

The purpose of this thesis project was to design and implement a system which maintained and processed electronic emitter data. The successful completion of this project resulted in a product which accomplishes the required tasks. This chapter will evaluate the utility of the techniques used during the course of the project. It also contains information on possible enhancements which might be desirable at a future date.

## Conclusions

The initial phase of the project was the requirements analysis, discussed in Chapter II. The technique used for this phase (Structured Analysis; see Appendix A and Ref 4) proved to be only partially successful. It was only used at a very high level. For larger projects where more communication is necessary among designers and with users, it would be more beneficial. It was useful as a means of organizing the ideas of the designer, but the user/designer interaction was accomplished without the data flow diagrams. Familiarizing the user with the technique would be more productive on a larger project.

The Structured Design technique (see Appendix A and Ref 8) used during the design specification stage was of greater value to this project. It proved to be a very organized and effective method of taking a general task and breaking it into modules which could be easily coded and maintained. Even small software projects such as this, can benefit from

this or similar techniques.

The choice of a programming language which encourages structured coding proved a wise decision. PASCAL is a highly readable language. While only an outside observer can determine how readable the resulting product is, the designer found debugging to be much simpler than expected.

A structured testing technique was employed on this project. The executive routines were tested first with "stubs" available as the system functions. Pieces were added systematically in a "top-down" manner. Organization of the testing in this manner also seemed to ease the debugging process. It also provided the designer with a more realistic appraisal of the progress of the project than a process of coding the entire program, then testing it all at once.

Recommendations

Since all the goals for the system were satisfied, recommendations for future system changes are probably not necessary or even desirable. Topics which might be considered by the user might include: the capability to automatically produce a problem tape based on the results of a search, the ability to dump the entire data base in update formatted card images as an input to a modified program (see Chapter VII), and a more efficient search algorithm once the system is converted to a new computer where access to permanent disk storage is possible. None of these are presently considered essential to the mission to be accomplished, but could be useful in a different environment.

Bibliography


1. Aho, Alfred V, John E Hopcroft, and Jeffrey D Ullman. The Design and Analysis of Computer Algorithms. Reading Mass: Addison - Wesley, 1974.

2. Baase, Sara. Computer Algorithms. Reading Mass: Addison - Wesley 1978.

3. Date, C.J. An Introduction to Database Systems. Reading, Mass: Addison - Wesley, 1977.

4. Demarco, Tom. Structured Analysis and System Specification. New York, New York: Yourdon Press, 1979.

5. Jensen, Kathleen and Niklaus Wirth. PASCAL User Manual and Report. New York, New York: Springer-Verlag, 1978.

6. Wiederhold, Gio. Database Design. New York, New York, McGraw Hill, Inc, 1977.

7. Wirth, Niklaus. Algorithms + Data Structures = Programs. Englewood Cliffs, New Jersey: Prentice-Hall, 1976.

8. Yourdon, Edward and Larry L. Constantine. Structured Design. New York, New York: Yourdon Press, 1975.

Appendix A

## Structured Analysis and Design Techniques

### Introduction

This appendix contains a brief explanation of the structured techniques used for this thesis. It is presented to provide the reader with a knowledge of the syntax necessary to interpret the diagrams in Chapters II and III. A thorough discussion of Structured Analysis and Structured Design is available in books by Denmaro (Ref 4) and by Yourdon and Constantine (Ref 8).

### Structured Analysis

This term is used to describe a top-down method of describing the requirements of a project. It uses a series of Data Flow Diagrams to set forth the tasks to be accomplished. It also uses tools such as decision tables and trees, and structured english to define the requirements of the system. Only the diagrams are present in this document.

The method begins by describing the system as a whole. Each major task of the system is shown on a diagram as a "bubble". The data to and from that bubble is indicated and defined in the following chart. Each upper level task is then broken into a set of subtasks on a lower level diagram. This process continues until each bubble on the lowest set of diagrams represents one particular process of the system. These usually have only one input and one output.

### Structured Analysis Syntax

Figure 45 illustrates the syntax of the technique. Data flows (control flows are not shown) are described as arrows. Processes are shown

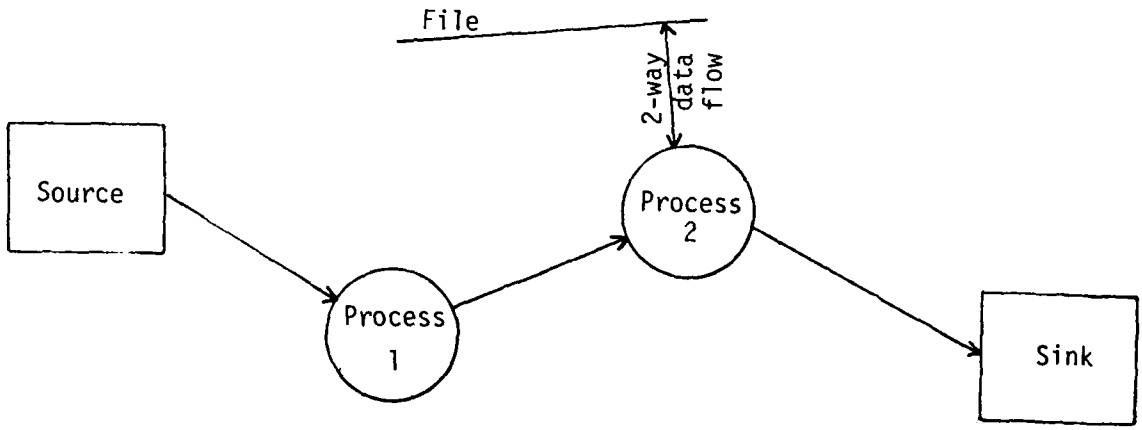



Figure 45. Structured Analysis Syntax

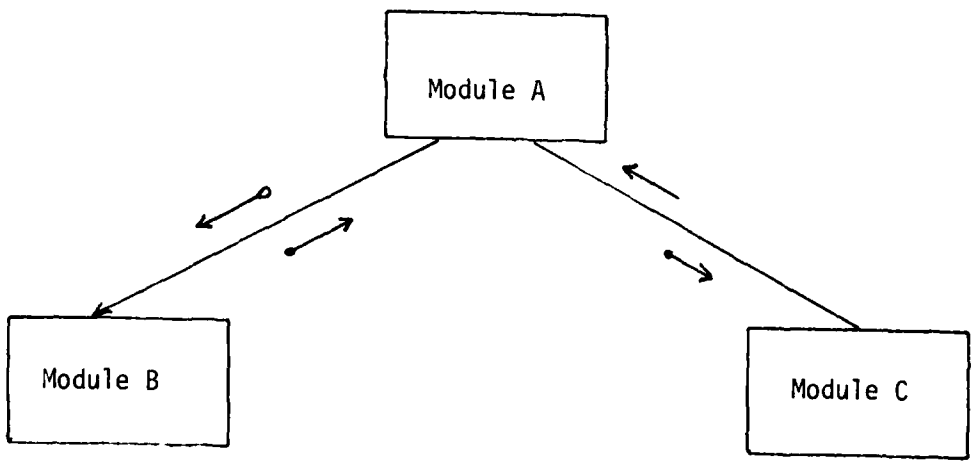



Figure 46. Module to Module Information Flow

as bubbles. Files are shown at the first level on which they are referenced and all references are shown at that time. The boxes represent sources or sinks for the data.

Structured Design

This term is used, in this document, to designate a technique for breaking the tasks to be performed into easily coded modules. It begins by describing the project as a bubble chart similar to those discussed above.

The bubble chart is then factored into a tree structure. The top-level block is called the coordinate module. Modules are broken down into a series of lower level modules. Whenever possible the modules are created to be one of three types; afferent (input), transform, or efferent (output). At higher levels these types are not always so clear cut. Each module is broken down into lower level modules until each represents a unit which performs one particular function.

The modules should be highly cohesive (i.e., a tight relationship between elements of a module) and minimally coupled (i.e., a loose relationship between modules). These modules should then be examined to ensure that each performs a useful function. If not, modules can be collapsed together to produce a useful module.

Structured Design Syntax

The syntax for the charts produced by this technique is illustrated in Figures 46, 47 and 48. The data and control flows are shown as the arrows on the charts. Note also the special symbols for decision and repetition.

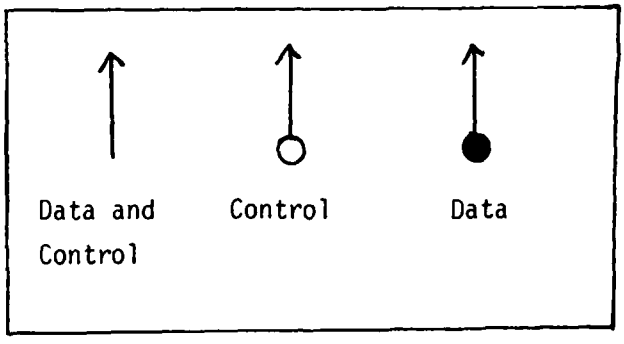



Figure 47. Information Arrows

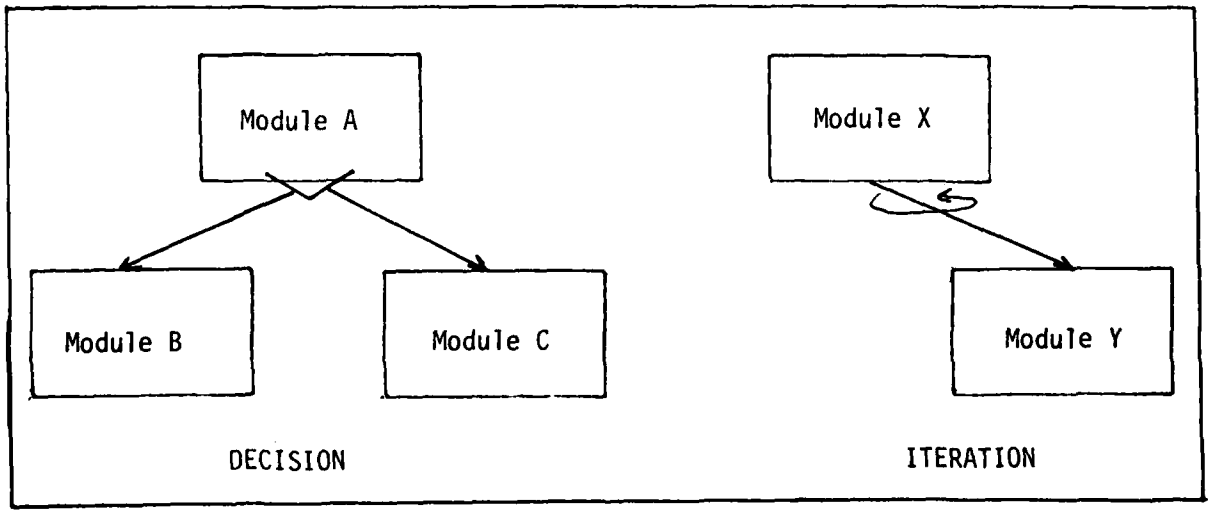



Figure 48. Decision and Iteration Syntax

Appendix B

User's Guide

## Introduction

This document contains the information necessary to use the system that has been designed to maintain and process electronic emitter data. It is organized in a manner which will allow it to be used independently of all other system documentation. It describes each of the system functions and provides information on the inputs necessary to use the system and the outputs which can be returned from the system.

Each of the system functions is described in a section below. In addition, the output selection process is provided as a separate section. Examples are used to illustrate realistic inputs and the resulting outputs. The sections are ordered in the sequence that would, most likely, be useful to a first time user. First, the update function would be used to create or bring up to date the emitter data file. Since both the manipulation functions (Search and Produce) use the output selection option, it is defined next. The search function allows the user to examine the data already present in the emitter data file and the produce function can then be used to build a problem tape which will be used as an input to the air defense models.

## General Comments

It should be emphasized that, with the exception of the data discussed in the update section, the control cards are free format. No particular card columns need be used for the system or function commands. In some cases, as detailed below, the inputs may be placed on one card with only a space delineating each control word.

The system functions may be requested in any order. Within some functions, the ordering of subfunctions is mandated, but the control for most pieces of the system is left in the hands of the user. The number of instances of each function in one system run is also up to the user. For example, a produce both before and after an update is possible.

Update

The update function is the emitter data file maintenance function. Emitter data can be inserted, changed, and deleted by this function. The order of the update subfunctions, Insert, Delete and Change is left to the user. The data is sequenced to be processed by emitter number, but the priority of duplicate update commands is generally determined by the sequence input by the user. For example, an existing record may be deleted by one update command and a new record with the same number inserted by another command. If the commands are input in the reverse order the delete command will delete the new record as well as the old record.

Input. The input to the Update Function consist of an update command (Figure 49), followed by one or more data cards. This packet can be followed by another update command and data cards. The update processing will continue until a system command or an EOF is encountered.

The DELETE control card is followed by data cards containing the emitter numbers of the records to be deleted. The numbers may be input one per card (Figure 50) or several per card (Figure 51) with a space separating the numbers.

The CHANGE and INSERT control cards are followed by a series of data cards as shown in Figure 49. The formats of these data cards is

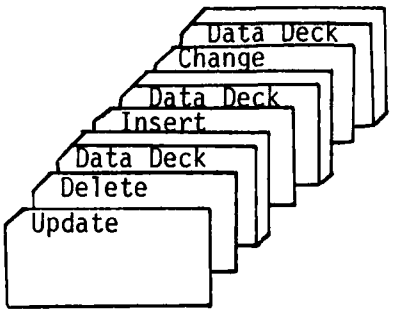



Figure 49. Update Control Cards

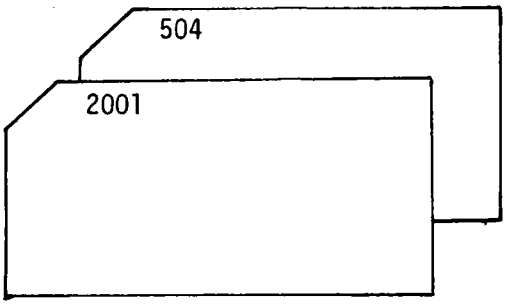



Figure 50. Delete Data Cards - One per Value Card

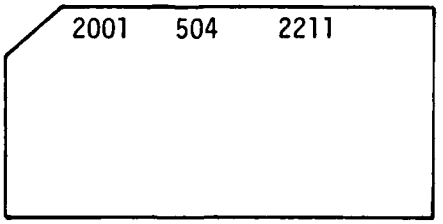



Figure 51. Delete Data Cards - Multiple Values per Card

fixed. It is based on the format of the existing card file for storing electronic emitter data and the input formats expected by the air defense models. The formats are shown in Table XLI and the input types listed in Table XLII. Each packet of update cards is terminated by another update subfunction, a system command, or an EOF.

Output. The output of an update function is a new electronic emitter data file. This file is automatically rewritten onto the input tape containing the old data file. In addition, error messages from the update function are returned to the user if the appropriate error condition exists.

Output Selection Option

Both the Search and Produce data retrieval functions allow the user to select which characteristics he desires to output. If the output selection option described in this section is not used, all characteristics will be output for the search function and all operational characteristics will be written to the problem tape during the produce function (i.e., the comment and classification data are the only characteristics not produced).

Input. The two output options are illustrated in Figure 52. The SELECT and SUPPRESS cards are free format. The Select option will provide an output of all characteristics contained on the data cards following. The legal output types are contained in Table XLII. The output types may be input one per card or several per card. The list of output types is terminated by a system control card or an EOF.

Search

The Search function queries the data file to produce a listing, describing the emitters which contain certain key parameters. The SEARCH

TABLE XLI

UPDATE Input and PRODUCE Output Format - Data

| Data Field | Column(s)* | Comments |
|---|---|---|
| Emitter Number | 1-5 | |
| Date | 7-11 | Date of source document |
| Field 1 | 13-20 | |
| Field 2 | 21-30 | |
| Field 3 | 31-40 | |
| Data Type Name | 41-50 | See Table XLII |
| Source Reference | 52-71 | |
| Blank, "C", or "S" | 73 | "C" indicates Comment, "S" indicates Classification |
| "D" | 74 | Designates Electronic Emitter Data Card |
| Data Type Number | 75-80 | See Table XLII |

*Undesignated Columns are blank.


Figure 52. Output Selection Control Cards

TABLE XLII

Legal Input/Output Data Types

| Data Type Number | Data Type Name | Data |
|---|---|---|
| 100 | SYSTM NAME | System Name |
| 110 | NRADAR | Short Name |
| 120 | TRANS NAME | Transmitter Name |
| 130 | RECVR NAME | Receiver Name |
| 140 | RECVR TYPE | Receiver Type |
| 200 | ELINT NO. | ELINT System Number |
| 220 | MAJOR SYS | Major System |
| 300 | NKODE | Index (Internal Model Use) |
| 310 | ECAC NO. | ECAC System Number |
| 400 | NFUNC | Code for Pulse Density Model |
| 410 | FUNCTION | ECAC Function Code |
| 600 | NBEAM | Number of Beams |
| 650 | IPPS | Average PRF |
| 660 | MODULATION | ECAC Code for Modulation |
| 700 | NPRF | Numper of PRF |
| 750 | PRF | Pulse Repetition Frequency |
| 800 | NPW | Number of PW |
| 850 | PW | Pulse Width |
| 900 | NRF | Number of RF |
| 950 | RF | Radio Frequency |
| 1000 | NSCAN | Number of Scanrates |
| 1050 | SCAN | Scan Time |
| 1100 | SCAN TYPE | |
| 1200 | WSCAN | Scan Width |
| 1300 | POWER | Transmitter Power |
| 1400 | NNOD | Number of Nodrates |
| 1450 | TNOD | Time for One Nod |
| 1500 | RTRACK | Max Detection Range |
| 1600 | TTRACK | Max Target Tracking Range |
| 1700 | WTRACK | Max Lethal Range |

TABLE XLII (Con't)

| Data Type Number | Data Type Name | Data |
|---|---|---|
| 1800 | WTRACK | Min Lethal Range |
| 1850 | ANTEN NAME | Antenna Name |
| 1900 | IANT | Antenna Pattern Type |
| 2000 | AGAIN | Antenna Gain |
| 2100 | HBW | Horizontal Beam Width |
| 2200 | VBW | Vertical Beam Width |
| 2300 | PHI | Tilt Angle |
| 2400 | POLAR | Polarization Code |
| 2500 | BEAM RF | Beam Radio Frequency |
| 8000* | CLASSIF | Classification (System) |
| 9000* | COMMENT | Comments (Emitter) |

*Not Legal Outputs for PRODUCE.

control card is followed by several packets, as shown in Figure 53. Each packet contains the key parameters and values or ranges which will be used to query the system. Each packet will result in a set of emitter characteristics describing the emitters possessing all the proper parameters or a message indicating that no qualifying emitters were found.

Input. The SEARCH control card begins the function processing. All following parameters are assumed to be the keys for the first query; until the first END card is encountered. The END card terminates one query. If the card following the END card is a FIN card, then the search processing begins. The FIN terminates the stream of packets. The number of queries is unlimited, as is the number keys used in a query (up to the number of legal values available).

The legal keys are specified in Table XLIII. They are input, one per card, with the alphanumeric designator obtained in Table XLII followed by a value or range of a type corresponding to the key parameter.

The FIN card may be followed by the output selection option described in the above section. If the option is not employed, the next system command may follow.

Output. The results of a query of the master emitter data file is a listing of the requested characteristics for qualifying emitters. The header lists the parameters used for this particular search. This hard-copy output should provide the user with his desired information along with source information such as date of last update, classification, and source document; as well as all comments. Error and information messages are also returned to the user.

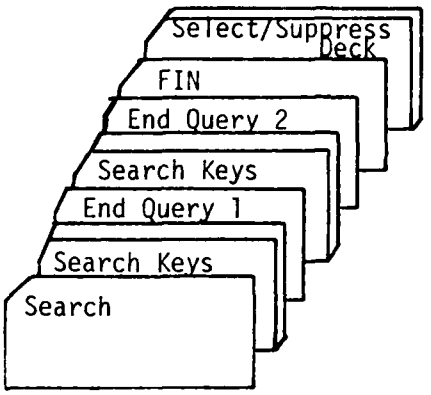



Figure 53. Search Control Cards

TABLE XLIII

Legal Search Keys

| Search Keys | Data Type/Stored As |
|---|---|
| IPPS | Integer/Value |
| PRF | Real/Range |
| PW | Real/Range |
| RF | Real/Range |
| SCAN | Real/Range |
| WSCAN | Real/Value |
| POWER | Real/Range |
| TNOD | Real/Value |
| RTRACK | Real/Value |
| TTRACK | Real/Value |
| WTRACK | Real/Value |
| XTRACK | Real/Value |
| IANT | Integer/Value |
| AGAIN | Real/Value |
| HBW | Real/Value |
| VBW | Real/Value |
| PHI | Real/Value |
| BEAM RF | Real/Value |

Produce

The Produce function builds a problem tape containing the electronic emitter characteristics of selected emitters. The tape contains card images of a fixed format which can be input to the simulation models selected and any combination of characteristics, for those emitters, can be placed on the tape.

Input. The PRODUCE control card begins the function processing. This command is followed by a series of emitter numbers (Figure 54). Each of these emitters will, if found in the data file, be processed for inclusion on the problem tape. The list of emitter numbers is terminated by a Select Output Option Card, a system control card, or an EOF.

Output. The results of the Produce function is a tape containing the card images selected by the user. The card images are of the format used by the update function (Table XLI). Error and information messages are listed for the user.

Additional Comments

This guide assumes the users are familiar with the operating system for the appropirate computer system. If the users are not familiar with commands to attach a tape file, as well as the job cards necessary to use the system in a batch node, they should refer to the System User's Manual.

Following conversion, this document should be updated to reflect changes necessitated by the change in system. The use of the VAX 11/780 system may allow enhancements which can be documented in this or a aimilar manual.

Figure 55 is a listing of a sample input deck that has been executed on this system. It exercises all the system functions and subfunctions.

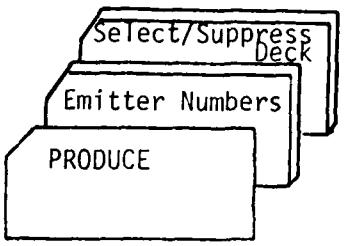



Figure 54. Produce Control Cards

```
UPDATE
 DELETE
    2001
    124   11372
 INSERT
6004  11/79   2   3.10            HBW       ECAC      D      2100
 .
 .
6004  11/79   1    .75   1.125    PW        ECAC      D      0850
 CHANGE
2106  11/79       110.0           WTRACK    ECAC      D      1700
 SEARCH
  PW   .990   1.05
  END
  RTRACK   300.0   500.0
  WTRACK   150.0   310.0
  END
  SELECT
    SYSTM NAME
    TTRACK
 PRODUCE
  6004
  2106
  121
  11371
 SUPPRESS
  ELINT NO.
```

Figure 55. Sample Control Stream

VITA

Clifford C. Gardner was born on 9 September 1952 in Montour Falls, New York. He graduated from high school in Watkins Glen, New York in 1970. He attended the State University of New York at Albany from which he received the degree of Bachelor of Science, Mathematics in May 1973. After completing Officer Training School, he was commissioned in the USAF in July 1974. Following training as a computer programmer at Sheppard AFB, Texas and Keesler AFB, Mississippi, he served as a programmer/analyst at the SAGE Programming Agency, Luke AFB, Arizona. He entered the School of Engineering, Air Force Institute of Technology in June 1978.

Permanent Address: 310 E 4th Street
Watkins Glen, New York 14891

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|---|
| 1. REPORT NUMBER<br>AFIT/GCS/EE/79-5 | 2. GOVT ACCESSION NO. | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE *(and Subtitle)*<br>AN EMITTER DATA MAINTENANCE AND RETRIEVAL SYSTEM | | 5. TYPE OF REPORT & PERIOD COVERED<br>MS Thesis<br>6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR*(s)*<br>Clifford C Gardner<br>Captain USAF | | 8. CONTRACT OR GRANT NUMBER*(s)* |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS<br>Air Force Institute of Technology (AFIT-EN)<br>Wright-Patterson AFB, Ohio 45433 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 11. CONTROLLING OFFICE NAME AND ADDRESS<br>Air Force Avionics Laboratory (AFAL/WRA)<br>Analysis and Evaluation Branch<br>Wright-Patterson AFB, Ohio 45433 | | 12. REPORT DATE<br>December 1979<br>13. NUMBER OF PAGES<br>149 |
| 14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)* | | 15. SECURITY CLASS. *(of this report)*<br>Unclassified<br>15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

Approved for public release; IAW AFR 190-17

JOSEPH P HIPPS, Major, USAF
Director of Information

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Software Engineering, Structured Design
Electronic Emitter Data Manipulation

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

The design and implementation of a system to automate the handling of electronic emitter data is documented in this report. Using such software engineering techniques as structured analysis and design, structured programming, and top-down testing, a system was produced which maintains data that will be used as inputs to certain analysis models. It also allows the user to automatically generate the input tape for the models and search the data file for emitters possessing desired key characteristics.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*